

Timi Rautamäki

DYNAMIC TRACKING OF PERSONNEL USING MACHINE VISION AND MACHINE LEARNING

Master's thesis
Faculty of Engineering and Natural Sciences (ENS)
Examiners: Professor Jose Luis Martinez Lastra
University Instructor Luis Enrique Gonzalez Moctezuma
March 2025

ABSTRACT

Timi Rautamäki: Dynamic tracking of personnel using machine vision and machine learning
Master's thesis
Tampere University
Master's Degree Programme in Automation Engineering
March 2025

Industrial shop floors present numerous hazards due to the interaction between personnel and machinery. Ensuring the safety of personnel in an industrial setting is crucial, but traditional safety mechanisms, such as RFID-based tracking and physical barriers, and predefined safety zones often require extensive setup and fail to dynamically adapt to changing conditions. These conventional methods also often fail to provide real-time information about potential safety issues, increasing the risk of life-threatening hazards. With the increasing complexity of industrial environments and the growing number of robots in shop floors, there is a growing need for intelligent, automated safety solutions that can adapt to dynamic conditions and provide real-time alerts for hazardous situations. This thesis studies the use of machine vision and machine learning for real-time personnel tracking, re-identification, and hazard detection to enhance industrial workplace safety.

A system is designed and implemented to detect, track, re-identify and ensure safety of personnel in an industrial setting using deep learning models. Object detection is performed using the YOLO framework, a fast state-of-the-art object detection model. Personnel tracking and re-identification utilize multi-object tracking algorithms and feature extraction techniques to maintain accurate identities across frames. The system also maintains a database of personnel to keep the feature extractions for re-identification up-to-date with possible changes in the visual environment and personal appearance. Additionally, the system integrates hazard identification through pose estimation and action recognition, allowing it to detect potential safety risks such as workers entering hazardous zones or appearing in distress. The integration of these components combines into a safety monitoring system that can operate in real-time without requiring specific hardware or extensive setup.

Experiments were conducted to evaluate the accuracy of personnel detection, tracking reliability under occlusion conditions, and re-identification robustness in different scenarios, including occlusion and overlapping of people. The results demonstrate that the proposed system achieves high detection accuracy while maintaining reliable tracking and re-identification. Furthermore, the system's hazard detection capabilities proved the system's potential to identify and mitigate workplace risks. The system was able to recognize safety threats such as unexpected falls and a person moving into a hazardous area. By reducing reliance on manual supervision and traditional workplace safety measures, the system presents an adaptable approach to industrial safety monitoring. The findings confirm the potential of machine vision-based tracking to improve safety in industrial environments.

Keywords: Machine vision, personnel tracking, re-identification, hazard detection, deep learning

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Timi Rautamäki: Henkilöstön dynaaminen seuranta konenäön ja koneoppimisen avulla
Diplomityö
Tampereen yliopisto
Automaatiotekniikan DI-ohjelma
Maaliskuu 2025

Teolliset ympäristöt, kuten tehtaot, aiheuttavat lukuisia potentiaalisia vaaratilanteita henkilöstön ja koneiden välisen vuorovaikutuksen vuoksi. Henkilöstön turvallisuuden varmistaminen on tärkeää, mutta perinteiset turvamekanismit, kuten RFID-pohjainen seuranta, fyysiset esteet ja ennalta määritetyt turva-alueet vaativat usein ympäristökohtaista asennusta, eivätkä ne pysty mukautumaan muuttuviin olosuhteisiin. Nämä perinteiset menetelmät eivät myöskään tarjoa reaaliaikaista tietoa henkilöstön turvallisuudesta, mikä lisää vakavien vaaratilanteiden riskiä. Teollisten ympäristöjen muuttuessa ja robottien lukumäärän lisääntyessä älykkäiden, automatisoitujen turvallisuusratkaisujen tarve kasvaa. Tällaiset ratkaisut voivat mukautua dynaamisiin ja vaihteleviin olosuhteisiin ja tarjota reaaliaikaisia hälytyksiä vaarallisista tilanteista. Tämä opinnäytetyö tutkii konenäön ja koneoppimisen käyttöä reaaliaikaisessa henkilöstön seurannassa, uudelleentunnistuksessa ja vaarojen havaitsemisessa teollisen ympäristön turvallisuuden parantamiseksi.

Järjestelmä on suunniteltu ja toteutettu havaitsemaan, seuraamaan, uudelleentunnistamaan ja varmistamaan henkilöstön turvallisuus teollisessa ympäristössä syväoppimismalleja käyttäen. Objektien tunnistus toteutetaan YOLO-mallin avulla, joka on nopea ja kehittynyt nykyaikainen objektintunnistumalli. Henkilöstön seuranta ja uudelleentunnistus hyödyntävät usean kohteen seuranta-algoritmeja ja henkilön piirteiden eristämistekniikoita, jotta henkilöiden identiteetit pysyvät muuttumattomina kuvasta toiseen. Järjestelmä myös ylläpitää tietokantaa henkilöstöstä, jotta uudelleentunnistusalgoritmin toimintavarmuus pysyy korkeana ympäristön olosuhteiden ja henkilön ulkonäön muutoksista riippumatta. Järjestelmä toteuttaa myös vaaratilanteen tunnistamisen asennon arvioinnin ja toiminnan havaitsemisen avulla, jolloin se voi havaita mahdollisia turvallisuusriskejä, kuten työntekijöiden siirtymisen vaaralliselle alueelle tai kaatumisen. Näiden komponenttien integrointi muodostaa turvallisuuden valvontajärjestelmän, joka toimii reaaliajassa ilman erityisiä laitteistovaatimuksia tai laajaa asennusta.

Järjestelmää testattiin arvioimalla henkilöstön tunnistuksen tarkkuutta, seurannan luotettavuutta henkilön ollessa osittain piilossa esteen takana sekä uudelleentunnistuksen toimivuutta erilaisissa skenaarioissa, mukaan lukien henkilöiden päällekkäisyys ja peittyminen. Tulokset osoittavat, että kehitetty järjestelmä saavuttaa korkean tunnistustarkkuuden säilyttäen samalla luotettavan seurannan ja uudelleentunnistuksen. Lisäksi järjestelmän vaaratilanteiden tunnistuskyky osoitti sen tunnistavan ja varoittavan turvallisuusriskeistä. Järjestelmä kykeni havaitsemaan turvallisuusuhkia, kuten odottamattomia kaatumisia ja henkilön siirtymisen vaaralliselle alueelle. Vähentämällä riippuvuutta manuaalisesta valvonnasta ja perinteisistä turvallisuusmenetelmistä järjestelmä tarjoaa mukautuvan lähestymistavan teolliseen turvallisuusvalvontaan. Löydökset osoittavat konenäköpohjaisen seurannan mahdollisuudet parantaa turvallisuutta teollisuusympäristöissä.

Avainsanat: Konenäkö, henkilöiden seuranta, uudelleen tunnistaminen, vaaran tunnistaminen, syväoppiminen

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

USE OF ARTIFICIAL INTELLIGENCE IN THIS WORK

Artificial intelligence (AI) has been used in generating this work:

- No
 Yes

I hereby declare, that the AI-based applications used in generating this work are as follows:

Application	Version
ChatGPT	4o

Purpose of the use of AI

In this thesis, AI was utilized in structuring, spell checking and helping with sentence formatting.

Acknowledgement of risks

I am aware that I am totally responsible for the entire content of the thesis, including the parts generated by AI, and accept the responsibility for any violations of the ethical standards of publications.

CONTENTS

1.	Introduction	1
1.1	Previous research	2
1.2	Research objectives	2
1.3	Research questions.	2
1.4	Outline	3
2.	Theoretical background	4
2.1	Machine vision.	4
2.1.1	Applications of machine vision	4
2.2	Convolutional neural networks	5
2.3	Object detection	8
2.3.1	Machine learning in object detection	8
2.3.2	State of the art of object detection	10
2.4	Multi Object Tracking in machine vision	14
2.4.1	Evaluation Metrics for multi object tracking	19
2.4.2	State of the art in multi object tracking	20
2.5	Re-identification of subjects in machine vision	22
2.5.1	Re-identification metrics	25
2.5.2	Temporal Feature Aggregation	26
2.6	Hazard identification in computer vision	27
2.6.1	Action detection	28
3.	Design.	30
3.1	Functional and non-functional requirements	30
3.1.1	Functional requirements	30
3.1.2	Non-functional requirements	31
3.2	Architecture	32
3.3	Object detection	35
3.4	Pose estimation	37
3.5	Action recognition	37
3.6	Personnel tracking	38
3.7	Real-time re-identification	39
4.	Implementation	41
4.1	Video capture	41
4.2	Implementation of object detection	41
4.3	Implementation of person tracking	43

4.4	Implementation of re-identification	43
4.4.1	Re-identification model training	44
4.5	Implementation of hazard identification	45
4.6	Implementation of the graphical user interface	47
5.	Results	48
5.1	Experiment setup.	48
5.2	Personnel detection & tracking accuracy	48
5.3	Re-identification accuracy	49
5.4	Hazard identification	50
5.5	Limitations	51
6.	Conclusions	52
6.1	Research questions.	52
6.2	Future possibilities	53
	References.	54

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
AP	Average Precision
API	Application Programming Interface
C2PSA	Stage Partial with Spatial Attention
C3k	Cross Stage Partial
C3k2	Cross Stage Partial with kernel size 2
CIoU	Complete intersection-over-union
CMC	Cumulative Match Characteristic
CNN	Convolutional Neural Network
COCO	Microsoft Common Objects in Context
CSP	Cross Stage Partial
CUDA	Compute Unified Device Architecture
DBT	Detection Based Tracking
DFT	Detection-Free Tracking
DL	Deep Learning
FC	Fully Connected
FLOPs	Floating-Point Operations per second
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
ID	Identity
IDF1	Identity F1 Score
IDFN	ID False Negatives
IDFP	ID False Positives
IDP	ID Precision

IDR	ID Recall
IDTP	ID True Positives
IoT	Internet of Things
IoU	intersection-over-union
LTCC	Long Term Cloth-Changing
MAP	Maximum a Posteriori
mAP	Mean Average Precision
ML	Machine Learning
MOT	Multi-Object Tracking
MOTA	Multiple Object Tracking Accuracy
MOTP	Multi Object Tracking Precision
NMS	Non-maximum suppression
OKS	Object Keypoint Similarity
ONNX	Open Neural Network Exchange
R-CNN	Region Based Convolutional Neural Network
ReID	Person re-identification
ReLU	Rectified Linear Unit
RFID	Radio-Frequency Identification
SPP	Spatial Pyramid Pooling
SPPF	Spatial Pyramid Pooling Fast
SSD	Single Shot MultiBox Detector
ST-GCN	Spatial Temporal Graph Convolutional Networks
TFA	Temporal Feature Aggregation
VTransE	Visual Translation Embedding
YOLO	You Only Look Once

1. INTRODUCTION

Industrial shop floors are high-risk work environments for personnel due to heavy equipment, hazardous materials, and electricity. Moreover, the number of machines is steadily increasing and currently, the number of robots is approximately 300 000 in the United States. On average, the number of robots has been increasing 9 % per worker annually. While the number of robots increases, the number of accidents related to robots is increasing. Between the years 2015 and 2022 there were 77 robot-related accidents in the United States, of which the majority were caused by stationary units. Mobile robots caused roughly a third of the accidents. In addition to robots, hazardous locations on the work site are a cause of workplace injuries. In Poland in the year 2012, there were more than 8 000 accidents caused by a falling object of which about half happened at an industrial setting. [1] [2]

In an industrial setting, where humans and machines operate together, manual supervision is often inadequate. Real-time monitoring of personnel through machine vision can improve the safety of people working with or near industrial machinery by allowing the machine to stop, or by warning a supervisor or the worker, in case a person is detected inside the safety area of the machine or another location that is deemed hazardous.

Traditionally, the safety of shop floor personnel has been guaranteed by limiting the workspace of the robot to not allow humans into the area that the robot occupies. Often, there are safety toggles in place that automatically stop the robot in case it is approached. For example, in case the worker crosses a specific line that is marked by a tape and observed by a laser system. Another example is to have a switch in the sliding door surrounding the robot's working area, that shuts down the robot the moment the sliding door is opened. There are also Radio-Frequency Identification (RFID) systems, that allow real-time tracking of personnel by detecting a worker carrying an RFID tag or by requiring an RFID tag to allow access to a certain area of the shop floor. While these approaches are functional, it requires extensive setting up and hardware and needs to be set up for each piece of machinery separately.

Recent studies have found great performance advancements in machine vision, especially in object tracking and object detection [3]. As an example, between versions v2 and v3, You Only Look Once (YOLO), a popular machine learning model for real-time object

detection, has seen an improvement from an average detection time of 43 seconds to only 2,5 seconds [3]. However, this increase in speed comes with a trade-off in accuracy. Newer versions of YOLO provide features such as pose detection, that can be used to determine what an individual is doing in a frame of a video stream. Furthermore, this data can be used to detect emergencies by utilizing algorithms that monitor the movement of the person's limbs. This enables the usage of the existing technology of object tracking to track personnel in real-time, which translates to generating useful information for necessary stakeholders in a timeframe that keeps the information meaningful.

The latest iteration of the Safety concept, Safety 4.0, aims at improving workplace safety by utilizing state-of-the-art technologies from Industry 4.0 that include Internet of Things (IoT), Artificial Intelligence (AI), machine vision and real-time analytics. It also includes wearable technologies such as sensor-embedded helmets and wristbands. The aim of Safety 4.0 is to proactively prevent workplace accidents and enhance risk identification. [4]

1.1 Previous research

Several studies on the topic of improving industrial workplace safety using computer vision have been conducted. One of the notable studies is introduced by Yange Li, et al., where a vision-based hazard identification system was developed for construction sites. The study consisted of creating a visual relationship detection and ontology. The system is able to produce sentence-based information of the personnel in the construction site, such as "worker wear helmet" and "person near robot". [5]. Another study by Hao Wu, et al. introduced an intelligent vision-based approach for helmet identification. [1]

1.2 Research objectives

The objective of this thesis is to implement an automatic personnel tracking system on a shop floor. The main purpose of this system is to recognize hazardous situations, such as personnel going towards machinery or robots, or medical emergencies. The system can also be used for recognizing the number of people on the shop floor and keeping track of people coming in and leaving the area. Tracking the entrance of personnel can be advantageous when they are working by themselves near robots or other industrial machinery with moving parts to verify that they are leaving the premises safely afterward.

1.3 Research questions

Tracking of personnel on a shop floor should be able to perform in real-time. Reliability with different illumination conditions is a key factor in the system. The system should also be able to perform Person re-identification (ReID) for people entering the view. Based on different conditions, such as a person going into a defined hazardous area or lying on the

ground for a specified time, the system will display the hazardous actions in a log. Given the above, the research questions this thesis answers to are:

1. How to optimize personnel tracking in real-time dynamic environments?
2. How to maintain the reliability of re-identification?
3. How to implement identification of hazardous situations?

1.4 Outline

This thesis consists of six chapters. In the second chapter, the theoretical background of the required technologies is covered. The third chapter covers the design of the system, and the fourth chapter covers the implementation of the system. The results are covered in the fifth chapter, and finally the conclusions and future possibilities are covered in the sixth chapter.

2. THEORETICAL BACKGROUND

This chapter introduces the main concepts for developing a machine vision system that implements machine learning. The following sections present relevant topics to this thesis. First, machine vision is defined followed by a section of Convolutional Neural Network (CNN) and Machine Learning (ML). Then, object detection is detailed, after which an overall picture of Multi-Object Tracking (MOT) is provided. The fifth section covers the topic of ReID and finally hazard identification is covered.

2.1 Machine vision

Figure 2.1 shows the typical principle of machine vision. In this setup, there is a computer, a camera, a target object, a light source, and a connection between the computer and the camera. The intended usage of the system changes the required hardware and software. The example system captures an image of the target object and sends it to the computer in a task called image acquisition. Finally, an action is triggered based on the result made during the image analysis by the computer software. Typically, computer software is used for image processing, but also a camera with built-in capabilities can be used. However, this approach is often limited in processing power. [6]

2.1.1 Applications of machine vision

Machine vision is widely used in industries such as manufacturing and security. A general purpose of machine vision in manufacturing is process control and quality assurance.

In manufacturing, the general use of machine vision is process control and quality assurance. A machine vision system could be used to sort different items based on a property, such as color, or scannable code. There are also possibilities in quality assurance. A machine vision system could also automatically determine if a manufactured piece appears correctly manufactured, and if not, either collect statistics or direct the faulty piece to another section in a manufacturing line.

Security applications for machine vision can range from a typical security camera setup that is capable of detecting movement in a video stream to a security robot to detect faces and notice environmental hazards such as smoke [7].

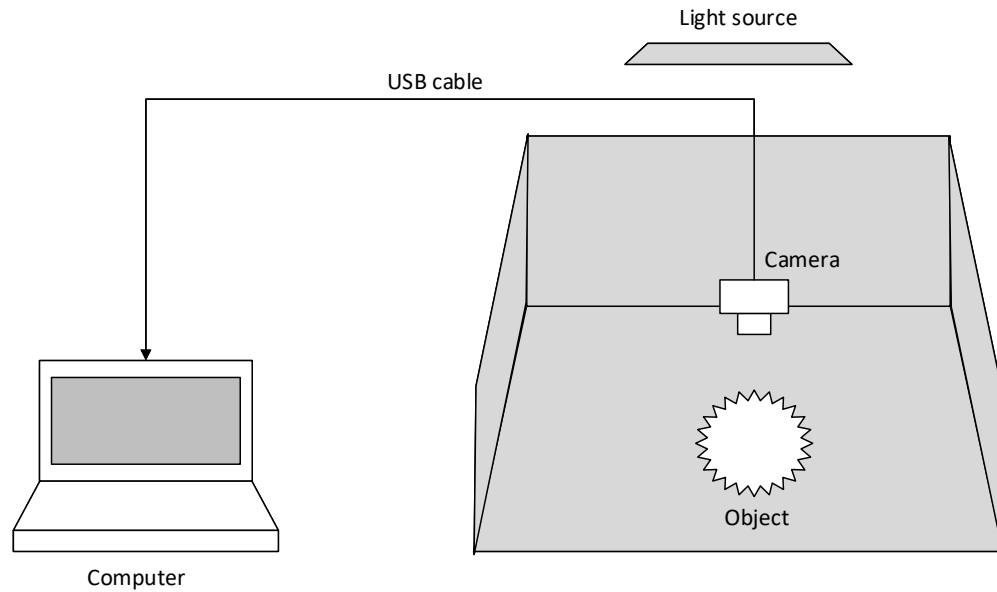


Figure 2.1. Principle of machine vision, adapted from [6]

2.2 Convolutional neural networks

A neural network is formed by multiple nodes or neurons. The nodes are connected with a value, that determines the weight of the connection. Each node calculates its output by calculating the weighted sum of each input and using an activation function. The activation function often aims to bring nonlinearity to the output, but also a linear activation function may be used that does not bring any nonlinearity. Having nonlinearity in the nodes allows the network to model more complex patterns. Common activation functions are *sign*, *sigmoid*, *tanh*, and Rectified Linear Unit (ReLU). A node is defined as:

$$\hat{y} = g\left(\sum_{j=1}^d w_j * x_i\right) \quad (2.1)$$

where \hat{y} is the output, g is the activation function, w_j is the weight of the connection and x is the input to the node. [8, p. 4–17]

A CNN is a neural network specialized in processing grid-shaped data. The data can be one-dimensional, for example in time-series data, or multidimensional, for example in pictures, where the grid cells represent color data in as many dimensions as needed. A grayscale image might only have one dimensional color data, whereas a colored image usually has three dimensions for color data. A convolutional neural network may have the same layers as a typical neural network, such as pooling and fully connected layers. A CNN is defined by using at least one convolutional layer, instead of a general matrix multiplication. An example of a CNN is shown in figure 2.2. [8]

The convolutional layer performs a convolution, that is defined as

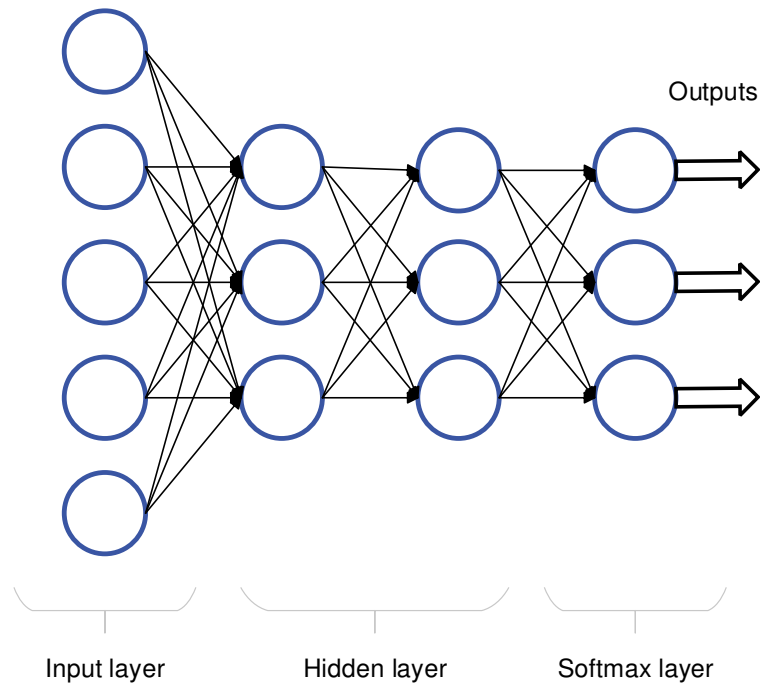


Figure 2.2. Layout of a CNN, adapted from [9]

$$s(t) = (x * w)(t) \quad (2.2)$$

where the function x is the input, function w is the kernel or filter of the layer, and the output s is referred to as the feature map. The convolutional layer produces a feature map by calculating a dot product between the input and the kernel in each position. The kernel is a matrix of weights that the neural network learns during training, and it is what produces the feature map. [8, p. 327–335]

A pooling layer is used to reduce the size of the input. A max pooling layer produces the maximum value as the output from a defined $n \times m$ submatrix of the input. It essentially lowers the dimensions of the input and in the case of an image, it lowers the resolution. An example of this layer is shown in figure 2.3, where the action is performed with 2×2 region with a stride of 2. This reduces the input resolution by a factor of 2. Similarly to max pooling, average pooling takes a defined size $n \times m$ matrix of the input and produces the average of the values. Pooling layers are important for a neural network as they reduce the parameter count and required computation while retaining the most critical information of the input. [8, p. 335–339]

Padding in CNN's is a technique to increase the output dimensions of a convolutional layer. In the case of images, padding adds extra pixels to the edges of the input. The purpose of this technique is to improve the network's performance near the edges of the image. Without padding, the very first row and column of pixels go through the operating layer only once. By adding padding, the features near the edges can be more reliably

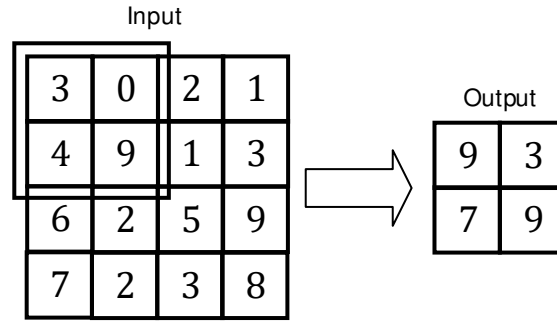


Figure 2.3. Max pooling layer

extracted as the layer passes the edge multiple times, similar to a pixel in the middle of the image. There are two main types of padding, valid padding and same padding. Valid padding means that padding is not used, and the filter does not exceed the edges of the image. Same padding means that padding is added to generate the same sized output as input, therefore the amount of padding depends on the kernel of the layer, assuming stride 1:

$$p = \left\lceil \frac{k - 1}{2} \right\rceil \quad (2.3)$$

for a filter size $k \times k$. [10]

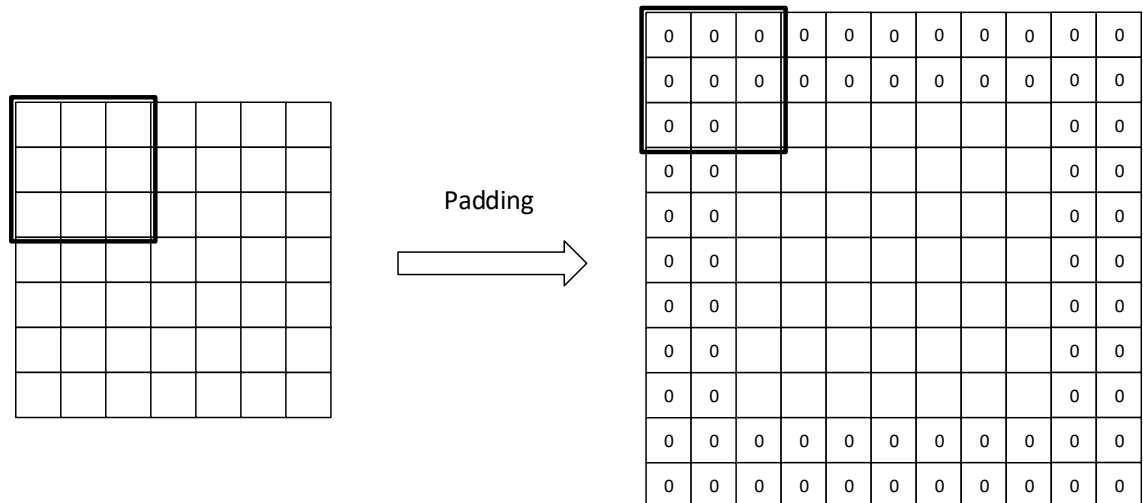


Figure 2.4. Example of zero padding

The main padding techniques are zero padding, replication padding, and reflection padding. Zero padding adds zeroes to the input image, as illustrated in figure 2.4. Replication padding extends the image by replicating the edge pixels to the padding [11]. Reflection padding adds reflected input values across the border axis [11]. [10]

2.3 Object detection

Identifying objects in an image is a generic problem in machine learning. Traditional methods for object detection include image color segmentation in different color spaces, such as RGB, HIS, and HSL, and using this data to identify objects given predefined constraints. These approaches are heavily affected by environmental conditions, specifically lighting. Given that the method involves extracting color data, for example, the target object being in a shadow greatly lowers the reliability of detections. Recently, the research has found great advancements in using ML for object detection, which is able to provide more accurate results with less input pre-processing. [12]

2.3.1 Machine learning in object detection

There have been numerous machine learning models intended for object detection, more recently Simple Linear Iterative Clustering, Support Vector Machine, and Random Forest [12]. Most recently, Deep Learning (DL) has become the state of the art in object detection [12]. Deep Learning provides object detection without any feature extraction from the input. There are many models in DL with continuous research providing improving results. Some recent models include Faster R-CNN (Faster Region-Convolutional Neural Network) and LEDNet.

The key evaluation parameters tested with object detection problems are:

- Precision: measures the reliability of the object detection result.
- Computational efficiency: the amount of system resources required for the process.
- Processing speed: the speed at which the results are available. [12]

The precision of object detection is evaluated by Mean Average Precision (mAP), which describes the accuracy of a model to detect objects. This property is defined using Precision P and Recall R , that are defined as follows:

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \quad (2.4)$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \quad (2.5)$$

where TP is true positives, FP is false positives and FN false negatives. Precision describes the percentage of correctly identified objects out of all predictions made and recall describes the percentage of positive predictions among all ground truths, or those that should have been predicted as positive. Therefore, to increase recall, the threshold can be lowered, which in turn lowers precision, as more false positives start to appear in

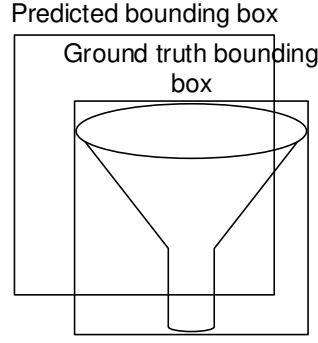


Figure 2.5. intersection-over-union (IoU)

the results.

True positives are defined using IoU. If the predicted IoU value exceeds a given threshold, the prediction is considered a true positive. The IoU is defined by the intersection of the predicted bounding box B_p and the ground-truth bounding box B_{gt} , divided by the union of these bounding boxes,

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (2.6)$$

as demonstrated in figure 2.5. Precision and recall values are used to create a precision-recall curve, that shows the drop in precision as the recall increases. This curve can be extrapolated and used to calculate the Average Precision (AP). Using 11-point interpolation, there are 11 fixed recall points with constant increments from 0.0, 0.1, ..., 0.9, 1 as seen in the equation 2.8. For each of these recall points, the interpolated precision is calculated as the maximum precision at that recall or any higher recall level, as shown below:

$$P_{interp}(R) = \max_{\tilde{R}: \tilde{R} \geq R} P(\tilde{R}) \quad (2.7)$$

where R is the recall point and \tilde{R} is the recall value. The AP is then computed as the average of these precision values at the recall points:

$$AP = \frac{1}{11} \sum_{R \in \{0, 0.1, \dots, 0.9, 1\}} P_{interp}(R) \quad (2.8)$$

Finally, the mAP is the mean average of AP values across all object classes:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.9)$$

where N is the number of evaluated classes and AP_i is the AP of the i th class. [13]

Computational efficiency and processing speed become important when object detection algorithms are used in real-time environments. A general way to determine computational efficiency is to measure the memory usage of the detector and the number of Floating-Point Operations per second (FLOPs). Usually models also report the number of parameters they use. Processing speed is measured in Frames Per Second (FPS), and it is a value that depends heavily on the processing hardware and the computational efficiency of the model.

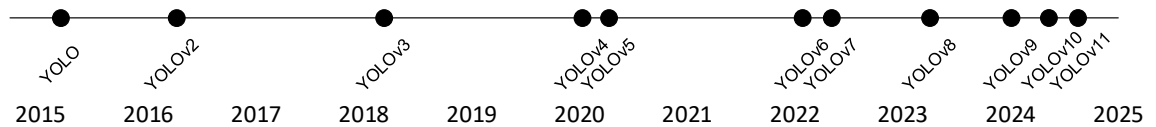


Figure 2.6. YOLO version history, adapted from [12]

2.3.2 State of the art of object detection

For the purpose of real-time object detection, the key parameters are computational efficiency and processing speed. With machine learning models, speed will be a trade-off with precision. Two of the outstanding machine learning models for object detection are YOLO and Single Shot MultiBox Detector (SSD). YOLO was initially released in 2015 by R. Joseph et al., as seen in figure 2.6. With its initial release, the key performance points showed a processing speed of 155 frames per second and mAP parameter at 52,7 %. YOLO version 1 also has an enhanced version that has a processing speed of 45 frames per second, but an improved mAP at 63,4 %. The SSD model was proposed by W. Liu, et al., also in 2015, and had a processing speed of 59 frames per second and mAP of 45,5 %. [14]

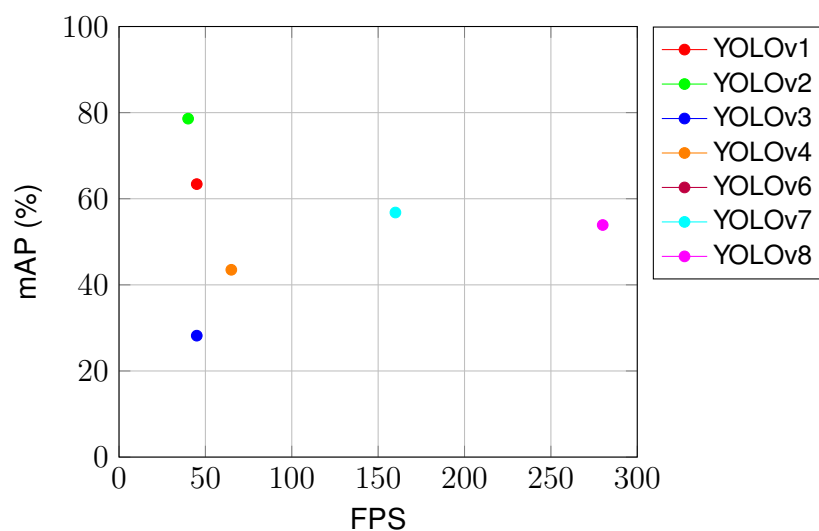


Figure 2.7. Progress of YOLO-model FPS and mAP

Figure 2.6 shows the release year timeline of YOLO-versions. The series has seen great changes over the years, even entirely changing the architecture in YOLOv4. Graph 2.7 shows the improvements between YOLO versions [15, 16, 17, 18, 19, 20, 21, 22]. Some noticeable drop in FPS can be attributed to the increased resolutions used in training and testing of the model, and the difference in testing hardware, and model level. Another factor to consider when making a comparison between the model versions is that the mAP would be smaller in case the image resolution used was also lower [16]. Therefore, this figure is only for reference to visualize the general direction of the YOLO models. The resolutions that the model uses for detection for each version are shown in table 2.1. YOLOv5 does not have a reviewed paper released on it and is therefore left out of this comparison. Similarly, versions v9, v10, and 11 do not include performance metrics in the released papers.

Table 2.1. *Input resolutions per YOLO-version.*

YOLOv1	YOLOv2	YOLOv3	YOLOv4	YOLOv6	YOLOv7	YOLOv8
224×224	544×544	320×320	512×512	640×640	640×640	640×640

As graph 2.7 shows, the main focus of the YOLO-system is real-time object detection at the expense of accuracy (mAP). Higher accuracies have been achieved with a significantly slower frame rate.

YOLO offers multiple models for each release since YOLOv5. For the latest release (11), five models are provided: nano (n), small (s), medium (m), large (l), and extra large (x). A comparison of the models is shown below in table 2.2. As can be seen, the smaller models have significantly fewer parameters and FLOPs at the expense of a lower mAP. However, the change in mAP is not increasing linearly between the models and the difference in mAP between medium and extra-large models is only 3,2 %.

The architecture of YOLO is presented in figure 2.8. The current base architecture of YOLO 11 models was formed in YOLOv8. It consists of three components, backbone, neck, and head. The backbone of the network is the feature extractor that consists of a CNN to output feature maps given images. The neck performs intermediate processing, its purpose is to enhance the feature maps. Finally, the head is the component that provides the predictions based on the enhanced feature maps.

Traditionally, anchor-based detectors have been the standard in object detection. In anchor-based detection, the image is split into a grid with many preset anchors, that serve as initial guesses for the positions and scales of the predicted bounding boxes. The anchors may be refined in a two-stage method, which achieves better results at the cost of speed. In anchor-free detection, the detector finds objects in two primary ways. First, by a keypoint-based method, where there are pre-defined or self-learned keypoints, using which the detector can predict the bounding box of the target object. The other method is

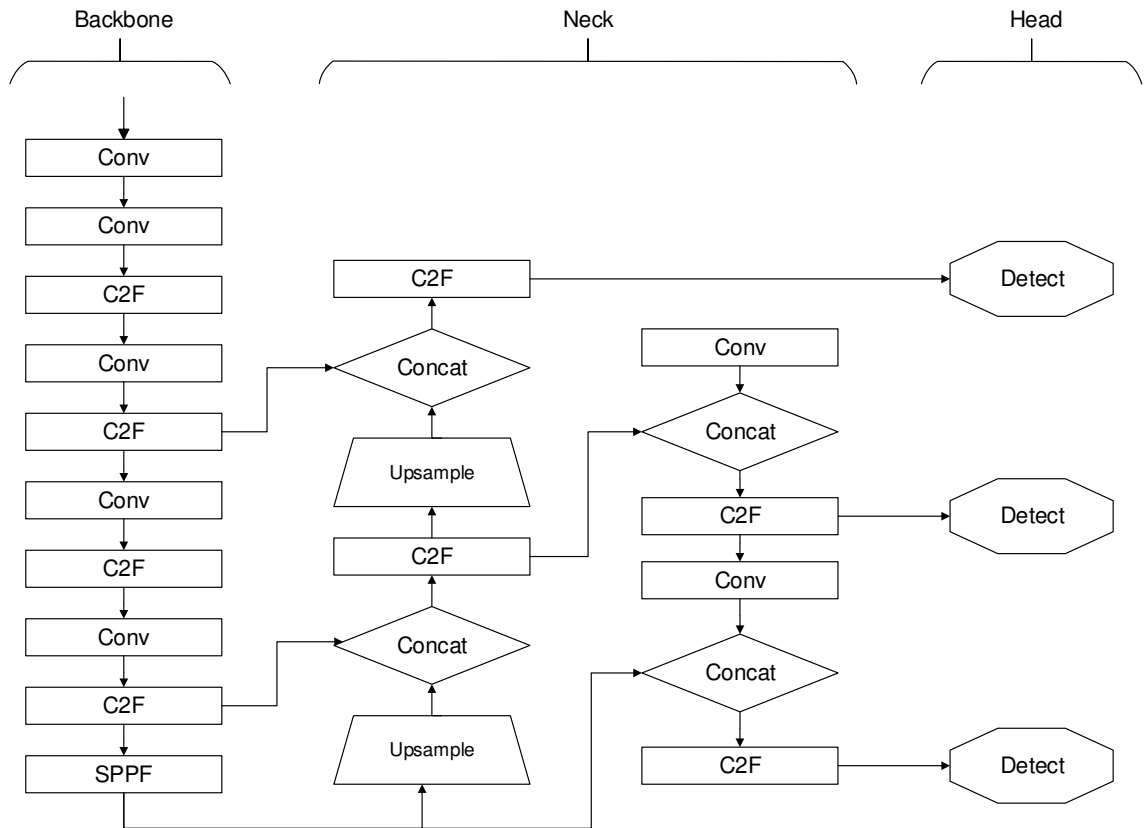


Figure 2.8. YOLOv8 architecture, adapted from [23]

called the center-based method, where the center point of an object is used to estimate the dimensions of the bounding box. Similarly to YOLOv8, YOLO11 uses anchor-free detection. [24]

Table 2.2. Comparison of YOLO11 models [25]

Model	Size (pixels)	mAP (50–95)	CPU ONNX (ms)	T4 TensorRT10 (ms)	Params (M)	FLOPs (B)
YOLO11n	640	39,5	56,1±0,8	1,5±0,0	2,6	6,5
YOLO11s	640	47,0	90,0±1,2	2,5±0,0	9,4	21,5
YOLO11m	640	51,5	183,2±2,0	4,7±0,1	20,1	68,0
YOLO11l	640	53,4	238,6±1,4	6,2±0,1	35,3	86,9
YOLO11x	640	54,7	462,8±6,7	11,3±0,2	56,9	194,9

The latest version of YOLO is version 11. It was released in September 2024 by Rahima Khanam and Muhammad Hussain. The improvements in this version are built on top of earlier YOLO versions, and the main innovations are the Cross Stage Partial with kernel size 2 (C3k2) block, the SPPF module, and the Stage Partial with Spatial Attention (C2PSA) block. The architecture of the backbone is illustrated in figure 2.9. YOLO version 11 replaced the C2f blocks found since YOLOv8 with C3k2 blocks. Finally, in the backbone, the Spatial Pyramid Pooling Fast (SPPF) block has remained unchanged, but

a new C2PSA block has been added after it. [26].

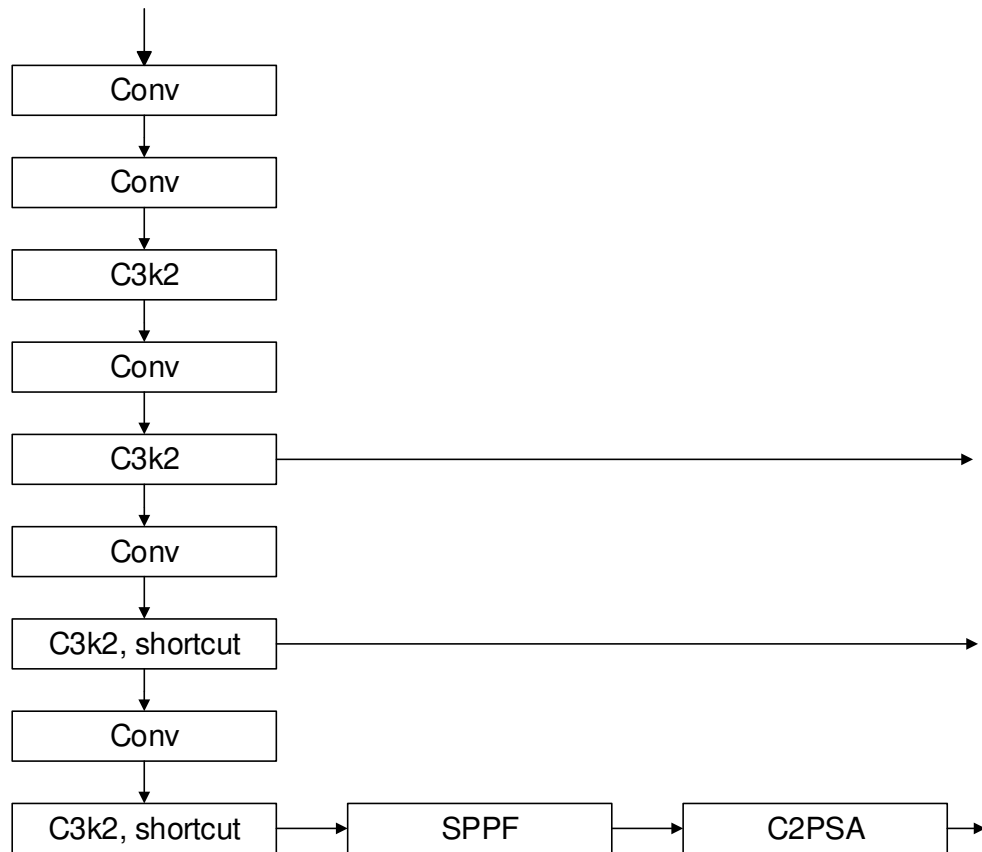


Figure 2.9. YOLO11 backbone, modified from [26]

The C3k2 block is a more efficient implementation of the Cross Stage Partial (CSP) bottleneck with 2 convolutions instead of one larger convolution, as it did in YOLOv8. The C3k2 block is illustrated in 2.10a. The block is a Cross Stage Partial (C3k) block with a kernel size 2, which is indicated by “k2” in the name. A C3k block is a flexible bottleneck module with a customizable kernel size. Both of the blocks are used for feature extraction. The C3k block contains n number of bottleneck-layers, with the parameter n depending on the used YOLO model size. The bottleneck layers also have the option to use a shortcut, also known as a skip connection or a residual connection, that is discussed in 2.5. [25]

The new C2PSA block in YOLO11 is found at the end of the backbone network in figure 2.9, with its task being to improve the focus of the key regions in the image. It consists of two parallel convolutional layers that are concatenated and input into a third convolutional layer. The C2PSA block is illustrated in figure 2.11a. The SPPF block is not newly introduced in YOLO11 but is an integral part of its backbone network architecture. The block is an enhanced version of the Spatial Pyramid Pooling (SPP) block, that behaves the same way, with the new SPPF block using less FLOPs and being faster, as the name indicates. The purpose of these blocks is to enhance the extracted features with different scales, sizes, and aspect ratios, which in turn improves the accuracy of the detector [27]. The SPPF block is illustrated in figure 2.11b. [26][28]

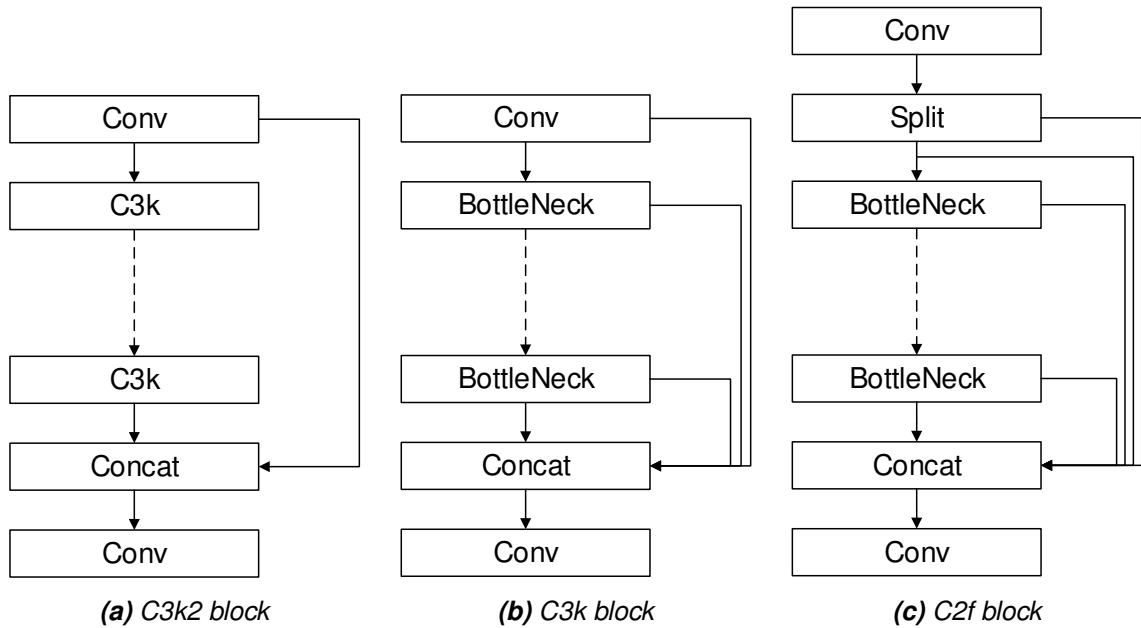


Figure 2.10. YOLO blocks, adapted from [25]

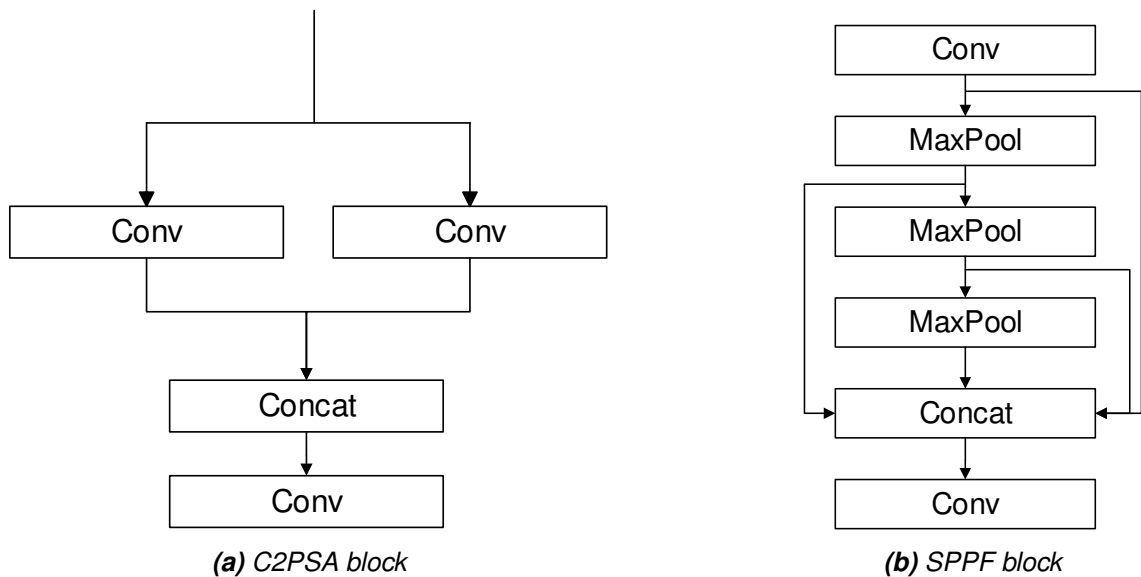


Figure 2.11. YOLO blocks, adapted from [25] and [29]

2.4 Multi Object Tracking in machine vision

Object tracking in machine vision is performed by MOT algorithms. MOT aims at following a target's movement in a video stream and keeping a static Identity (ID) for that object, instead of generating a new ID for each frame by only using a machine learning algorithm to detect the objects. This principle is shown in figure 2.12, where the identification for the objects is illustrated by the color of the bounding box. MOT algorithms typically interact with the output of a deep learning detector, usually processing the bounding boxes, as seen in Detection Based Tracking (DBT). Less commonly, they may work directly with the data from the camera's sensor, as done in Detection-Free Tracking (DFT).

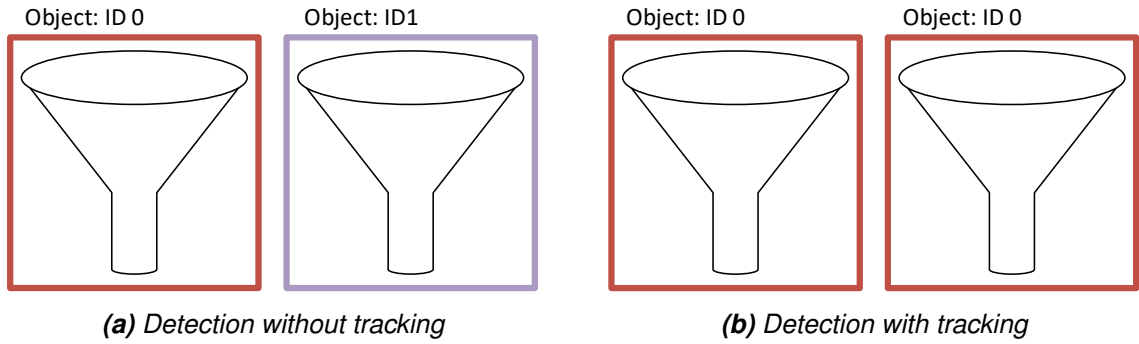


Figure 2.12. Example of object tracking

In MOT, each object has a state that generally describes the object's position and properties for motion. Other values that can be included in the state are other properties of the target, such as size or color. MOT is generally viewed as a multi-variable estimation problem. For a multi-variable estimation problem in a video stream, the states are defined as:

$$S_t = (s_t^1, s_t^2, \dots, s_t^{M_t}) \quad (2.10)$$

where t is the frame, i is the object ID and M is the number of objects, that describes the states of the objects M_t in the t th frame. Therefore, there is defined also

$$s_{i_s}^i = \{s_{i_s}^i, \dots, s_{i_e}^i\} \quad (2.11)$$

that shows the sequential states of the i th object. Here i_s is the start frame, and i_e is the end frame. For all the sequential states of all the objects, there is

$$S_{1:t} = \{S_1, S_2, \dots, S_t\}. \quad (2.12)$$

As above, in equation 2.10, the detections for the i th object in the t th frame is defined as

$$O_t = (o_t^1, o_t^2, \dots, o_t^{M_t}) \quad (2.13)$$

where o_i^t is the detection of the i th object in the t th frame. Also,

$$O_{1:t} = \{O_1, O_2, \dots, O_t\}. \quad (2.14)$$

similarly, as in equation 2.12, $O_{1:t}$ describes detections of the objects from the first frame to the t th frame. This defines MOT as a Bayesian statistics problem, which is defined by Maximum a Posteriori (MAP) estimation of the sequential states given the detections:

$$\hat{S}_{1:t} = \arg \max_{S_{1:t}} P(S_{1:t} | O_{1:t}) \quad (2.15)$$

which is finding the most likely state of each object in all frames. Equation 2.15 is what the MOT algorithms try to solve using different approaches. [30]

The two main initialization methods for MOT algorithms are DBT and DFT. DBT works by first feeding the image from the video source to the detector and then using the output of the detector as input to the tracker algorithm. Generally, the tracker needs the bounding box for each detection that is used for creating a status s for each object. DFT requires manual initialization of the objects, and it cannot work with appearing or disappearing objects. This makes the DBT initialization method more flexible for general use cases but requires pre-training of the detector. DFT on the other hand achieves greater computational performance, as it does not require a resource demanding detector. [30]

There are two processing modes for MOT algorithms, online and offline tracking. In online tracking, the sequence is handled frame by frame, providing information from the previous frame to the frame currently being processed. Offline tracking is used for existing video material, where the future frames can also be used as a reference for the current frame. Offline tracking is often more accurate, but it does not work in a real-time scenario. [30]

Finally, there are two types of outputs, stochastic and deterministic. Stochastic tracking uses probabilities to account for occlusion and other uncertainties in the object's movement. These trackers use a probabilistic model to estimate the state of the object, such as the Kalman Filter. Deterministic trackers don't take uncertainties into account. These trackers use fixed rules for tracking, such as the Hungarian algorithm. Deterministic tracking is simpler and faster, but stochastic tracking is often needed for real-world situations with uncertainties and occlusion. [30]

Kalman Filter

The Kalman filter was introduced in the 1960s by Rudolf Emil Kalman to predict the state of an uncertain dynamic system. The first derivation of the filter defines

$$X_{[k+1]} = AX_{[k]} + BU_{[k]} + GV_{[k]} \quad (2.16)$$

$$Y_{[k+1]} = CX_{[k+1]} + W_{[k+1]} \quad (2.17)$$

where $X \in \mathbb{R}^n$ is the state vector, which can be linked to equation 2.12, U the process input, and $Y \in \mathbb{R}^m$ measurement of the state. V is the model uncertainty and W noise from measurement uncertainty and digitalization, both V and W are independent

of each other. The Kalman Filter is used to give an estimate of the state X , when the measurement Y does not allow for a full understanding of the state of the object. [31]

The Kalman Filter has two main steps: prediction and update. In the prediction step, the previous state X and control variable A are used to estimate the object's next state $X_{[k+1]}$ with the noise V . This is called the priori state prediction [32]. In the update step, the predicted state $X_{[k+1]}$ is updated using the measurements Y that contain the measurable parts C of the object's state, and noise W . These steps are illustrated in figure 2.13. [31]

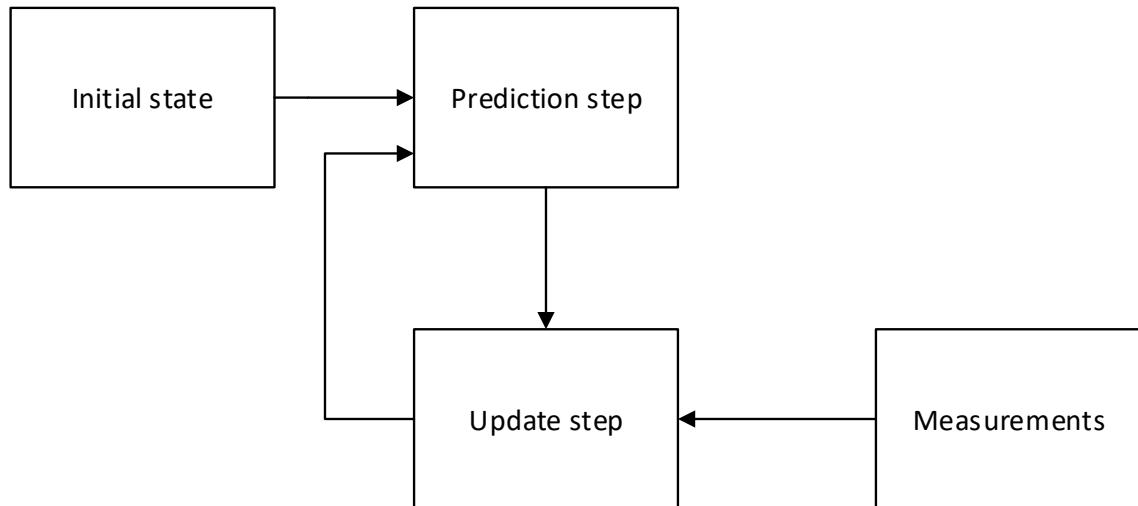


Figure 2.13. Kalman filter operations, adapted from [33]

The process noise V and measurement noise W are assumed to be independent and distributed according to normal probability [34]. Both V and W have covariance matrices Q and R respectively [34]. The noise and balance between measurement and prediction at $k + 1$ are linked by the Kalman gain K_{k+1} :

$$K_{k+1} = P_{k+1}C^T(CP_{k+1}C^T + R)^{-1} \quad (2.18)$$

here, P_k is the error covariance matrix of the prediction, C is the measurement matrix, and R is the measurement noise covariance matrix. If the process has a large noise in the measurements, the filter prefers to trust the prediction more. Similarly, if the prediction has a strong noise, it prefers the measurement more. [32]

Hungarian method

The Hungarian method is an optimization algorithm that is developed to solve an assignment problem. In reference to MOT, the assignment problem is linking objects in the current frame to the same objects in the previous frame [32]. The algorithm was introduced by Kuhn, H. W. in 1955 [35]. Instead of going through all the possible combinations in a cost matrix C , the algorithm is able to find the solution faster, specifically in compu-

tational complexity $O(n^3)$ [36]. In a matrix formulation, the Hungarian algorithm can be defined as

$$\min_P Tr(PC) \quad (2.19)$$

where C is a cost matrix and P is a permutation matrix. Each cell in C represents the cost of assigning a detection in the previous frame to a detection in the current frame. This means that if the object is visually similar to the object across two frames, the cost of the match is low. Otherwise, the cost is high.

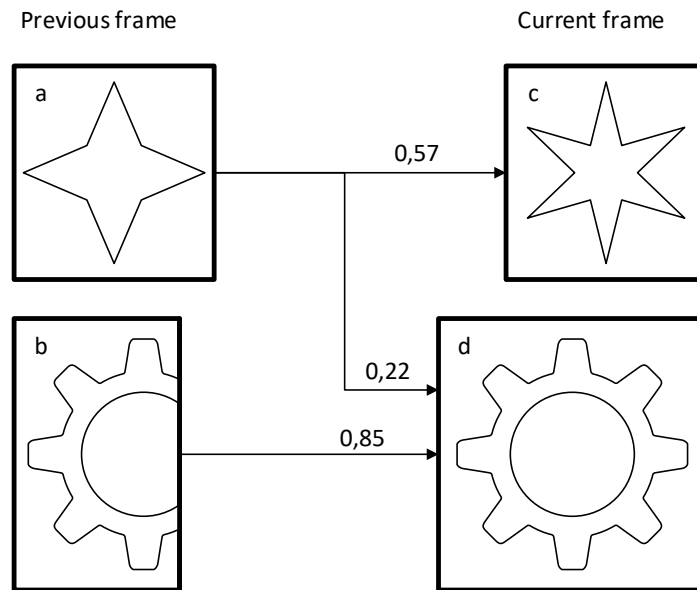


Figure 2.14. Hungarian matching example

The Hungarian algorithm in reference to MOT is illustrated in figure 2.14. In the figure, on the left side, there are previously detected shapes, with shape b being occluded partly. In the next frame on the right, the same shape d is entirely visible. Here, the probability that the shape is the same as the previous one is estimated to be 85 %. In addition, the top shape a is estimated to be matching the current frame objects with a probability of 57 % for c and 22 % for d . These values can be written in a matrix:

	c	d	
a	0,57	0,22	(2.20)
b	0	0,85	

Performing the Hungarian method step by step:

1. Subtract row minima:

$$\begin{array}{cc}
 & \begin{array}{cc} c & d \end{array} \\
 \begin{array}{c} a \\ b \end{array} & \begin{array}{cc} 0,57 - 0,22 & 0,22 - 0,22 \\ 0 - 0 & 0,85 - 0 \end{array}
 \end{array} \quad (2.21)$$

$$\begin{array}{cc}
 & \begin{array}{cc} c & d \end{array} \\
 \begin{array}{c} a \\ b \end{array} & \begin{array}{cc} 0,35 & 0 \\ 0 & 0,85 \end{array}
 \end{array} \quad (2.22)$$

2. Subtract column minima:

$$\begin{array}{cc}
 & \begin{array}{cc} c & d \end{array} \\
 \begin{array}{c} a \\ b \end{array} & \begin{array}{cc} 0,35 - 0 & 0 - 0 \\ 0 - 0 & 0,85 - 0 \end{array}
 \end{array} \quad (2.23)$$

$$\begin{array}{cc}
 & \begin{array}{cc} c & d \end{array} \\
 \begin{array}{c} a \\ b \end{array} & \begin{array}{cc} 0,35 & 0 \\ 0 & 0,85 \end{array}
 \end{array} \quad (2.24)$$

3. Cover all zeros with a minimum number of lines: All zeroes can be covered using 2 lines, while also $n = 2$, therefore matrix 2.24 is the optimal solution.

The Hungarian algorithm would efficiently match these shapes by minimizing cost or maximizing the probabilities. Following the steps, it can be seen that shape a is likely to be same as shape c , and shape b is likely to be shape d . [37]

2.4.1 Evaluation Metrics for multi object tracking

Evaluation of MOT is challenging, but multiple methods exist to show the performance of a MOT system in numbers [38]. CLEAR MOT metrics are used in official MOTChallenge benchmarks to evaluate MOT performance [39]. The CLEAR MOT metrics include Multiple Object Tracking Accuracy (MOTA) and Multi Object Tracking Precision (MOTP), that are the primary metrics used for evaluating the performance. MOTA is defined as

$$\text{MOTA} = 1 - \frac{\text{FN} + \text{FP} + \theta}{\text{GT}} \quad (2.25)$$

where False Negative (FN) is count of FNs, False Positive (FP) is count of FPs, and θ is count of identity switches. MOTA represents the overall accuracy of the tracker by

penalizing all errors relative to the ground truth GT . The MOTP metric identifies false positives, misses, and incorrect identity switches [38]. One downside of this metric is that the identity switch is penalized only once regardless of the duration of the misidentification, therefore even a one-frame error is considered as large as an error continuing for any number of frames. MOTP is defined as

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \quad (2.26)$$

where c_t is the number of matches in frame t and $d_{t,i}$ is the overlap between the predicted and ground truth bounding boxes [40]. MOTP describes the system's performance on estimating the detected object's position [38].

Other relevant key performance indicators are Identity F1 Score (IDF1), ID Precision (IDP) and ID Recall (IDR) [38]. These metrics were introduced by DukeMTMC benchmark [39]. IDF1 is defined as

$$\text{IDF1} = \frac{2 * \text{IDTP}}{2 * \text{IDTP} + \text{IDFP} + \text{IDFN}} \quad (2.27)$$

which shows the accuracy of correct detection over the average number of ground truth detections and computed detections. IDF1 depends on ID True Positives (IDTP), ID False Positives (IDFP) and ID False Negatives (IDFN), that are defined as

$$\text{IDFN} = \sum_{\tau} \sum_t m(\tau, \gamma_m(\tau), t, \Delta) \quad (2.28)$$

$$\text{IDFP} = \sum_{\gamma} \sum_t m(\tau_m(\gamma), \gamma, t, \Delta) \quad (2.29)$$

$$\text{IDTP} = \sum_{\tau \in AT} \text{len}(\tau) - \text{IDFN} \quad (2.30)$$

where τ is the true trajectories, and γ is the computed trajectories [39]. IDTP stands for correctly identified detections, IDFP stands for an incorrect tracking prediction, and IDFN stands for objects that should be tracked, but are not. $m(\tau, \gamma, t)$ calculates the number of missed detections between frames τ and γ [39].

2.4.2 State of the art in multi object tracking

Recent advancements in MOT have seen new algorithms such as FairMOT, CTrack, and SOTMOT. A comparison of MOT algorithms is shown in figure 2.15. In the figure, the key performance indicators are MOTA and FPS.

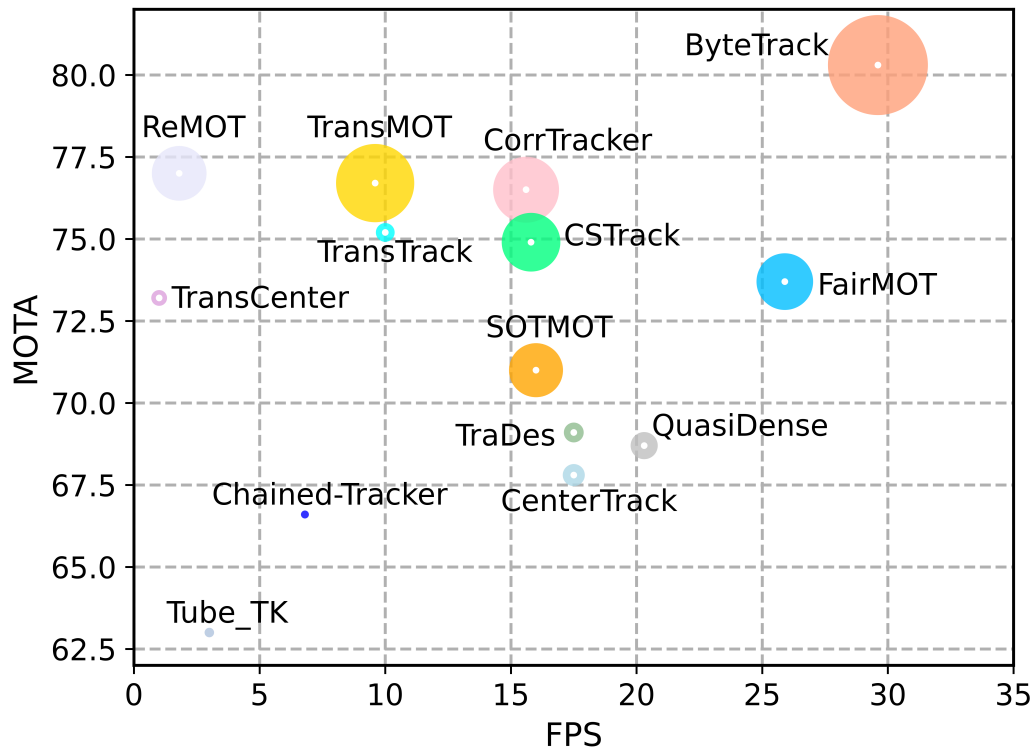


Figure 2.15. Comparison of MOT systems [41]

A recent MOT state-of-the-art algorithm is ByteTrack, developed by Zhang, Yify, et al. The problem with most previous multi-object trackers is that they do not account for the low score detections, which may happen when objects are occluded, the lighting, or other environmental variables change. This caused a new detection after the interference disappeared. The aim of ByteTrack is to address the issue with low confidence detections and therefore improve the robustness of the tracking. This is an especially important feature in dynamic environments, where objects can occlude each other, they can disappear partly behind other objects or be in low visibility due to lighting or other conditions. If these objects were filtered, the whole tracking algorithm would fail, as any object not confidently identified by the object detector would be dismissed. In addition to the key points in figure 2.15, the algorithm performs well in other metrics as well: the IDF1 value is 79,3 % without ReID and 80,5 % with re-id, which is more than 2 % better than compared algorithms MOTDT, DeepSORT, and SORT. [41]

ByteTrack is categorized as an online, stochastic, DBT algorithm. It takes the following configuration parameters:

- Track threshold: Threshold for determining if to track a detection or not based on the detection score.
- High threshold: Threshold for determining whether a new track is created for an

object, or if it is passed for this frame.

- Match threshold: Threshold for matching existing tracks.
- Track buffer: Buffer length to keep old tracks.

ByteTrack takes the detections as input, formatted as an object that contains the bounding box parameters and the detection score. Then it compares each detection score:

- Items that have detection score higher than the threshold: $\text{detection score} > \text{track threshold}$. These results are stored in a list `D_tracks`.
- Items that have detection score lower than the threshold: $\text{detection score} < \text{track threshold}$. These results are stored in a list `D_low_tracks`.

For each object, Kalman filter is used to predict a new location. Then, the first association is run by comparing the objects and those placed in the list `D_tracks` based on IoU. The unmatched results are stored in a list `remain_D_tracks` and `remain_T_tracks`. After this, the second association is run by comparing the lists `D_low_tracks` and `remain_T_tracks`. The unmatched results from this association run are stored in `current_lost_tracks`. The unmatched tracks in `current_lost_tracks` are stored for a defined duration `track_buffer`, often 30 frames. Finally, the new tracks are created from `remain_D_tracks`, if the object passes the `high_threshold`. [41]

2.5 Re-identification of subjects in machine vision

ReID is the task of recognizing the same individual after disappearing from the video stream. This involves remembering the appearance of the individual and later assigning the same ID for the person. ReID covers two situations:

- Short term ReID: re-identifying objects after being occluded for a short moment based on the appearance of the object or based on the movement pattern and calculated probabilities. This is usually done by a MOT algorithm, that utilizes a tracking algorithm, such as Kalman filter, or an assignment algorithm, such as the Hungarian method.
- Long term ReID: re-identifying objects after an extended period. This generally involves extracting more object-specific features, such as person clothing and appearance, that are used for deep learning algorithms to match the objects later. Long-term ReID can also support Long Term Cloth-Changing (LTCC), which has been the focus of research by Qian X. et al. [42].

ReID can be categorized either as a classification problem or a verification problem in machine learning. In a classification problem, a label, in this case, the person ID, needs to be predicted for a given input. In a verification problem, two images of the subject are compared, and it is verified whether they represent the same subject or not.

Table 2.3. ResNet comparison, adapted from [45]

Layer name	Output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2.x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3.x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4.x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5.x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1,8 * 10 ⁹	3,6 * 10 ⁹	3,8 * 10 ⁹	7,6 * 10 ⁹	11,3 * 10 ⁹

The general way of solving this problem is to first detect the subject, extract its features, and finally match the features to existing data. The ReID therefore happens after detection is performed, but before the MOT algorithm. Also, the MOT algorithm needs to be modified to use IDs found by the ReID algorithm. The recent methods of ReID have progressed to CNN, and Generative Adversarial Network (GAN) [43].

The main challenges in ReID are low resolution, light changes, pose changes, and occlusion [43]. However, according to Wei W. et al., the challenges of pose changes, illumination changes, and occlusion can be overcome by a good enough feature learning algorithm, and challenges with the perspective of the subject can be overcome with a robust metric learning algorithm [43].

Generally, any backbone designed for detection tasks, such as ResNet-50, can be used in ReID with minimal modifications, and it should be trained with a ReID dataset, such as Market-1501. The important modifications for ResNet-50 are changing the last convolutional layer and adding average pooling as the last pooling layer [44].

ResNet

ResNet stands for a Residual Network, and it was originally developed by Kaiming He, et al. in 2015. It solves the problem of degrading accuracy as the neural network's depth increases by introducing a skip connection, also called a residual connection. This allows the output of a neural network layer to be used as input to multiple layers later in the chain, as opposed to output-input being sequential. A partial architecture of ResNet-50 with the skip connections is illustrated in figure 2.16b. ResNet provides multiple different versions, each varying in the number of parameter layers. A comparison of the versions can be seen in table 2.3. [45]

For example, ResNet-50 is a deep convolutional neural network with 50 layers in multiple stages. The layers are:

- Conv1: a 7×7 convolutional layer with 64 filters and stride of 2.
- Conv2.x:
 - 3×3 max-pooling layer.
 - 1×1 convolutional layer.
 - 3×3 convolutional layer.
 - 1×1 convolutional layer.
- Conv3.x, Conv4.x, Conv5.x
 - 1×1 convolutional layer.
 - 3×3 convolutional layer.
 - 1×1 convolutional layer.
- Average pooling

Here, the max pooling layer is defined with 3×3 , and a stride of 2. This means that for each 3×3 area of the input to this layer, the maximum value is taken and forwarded into the output of this layer. The stride of 2 causes some overlap, as the layer reads data by moving 2 cells after each read.

Each of the convolutional layers within the ResNet-50 network contains similar convolutional blocks, with differing number of residual blocks and amount of filters. The final layer of the network is a fully connected layer following average pooling. The average pooling layer essentially prevents overfitting by simplifying the feature map from the previous layer. Its input is $7 \times 7 \times 2048$ and it outputs a $1 \times 1 \times 2048$ vector that is the input of the fully connected layer. The fully connected layer has as many neurons as there are output classes. In each neuron, the layer produces scores that represent the probability of each of the classes, 1000 in this example. The final output is formed by the softmax activation function, which converts the scores into probabilities to determine the predicted output of the network. [45]

Market-1501

The Market-1501 dataset was introduced by Zheng, Liang et al. in 2015. The dataset contains 32 668 bounding boxes of 1 501 different people. Each person is captured by at least two cameras, which ensures that different angles of the subject are captured. The dataset also introduces imperfections instead of perfect bounding boxes, causing missing parts and misalignment of the bounding boxes. Finally, the dataset also contains false alarms by marking bounding boxes as 'good', 'distractor', or 'junk'. [46]

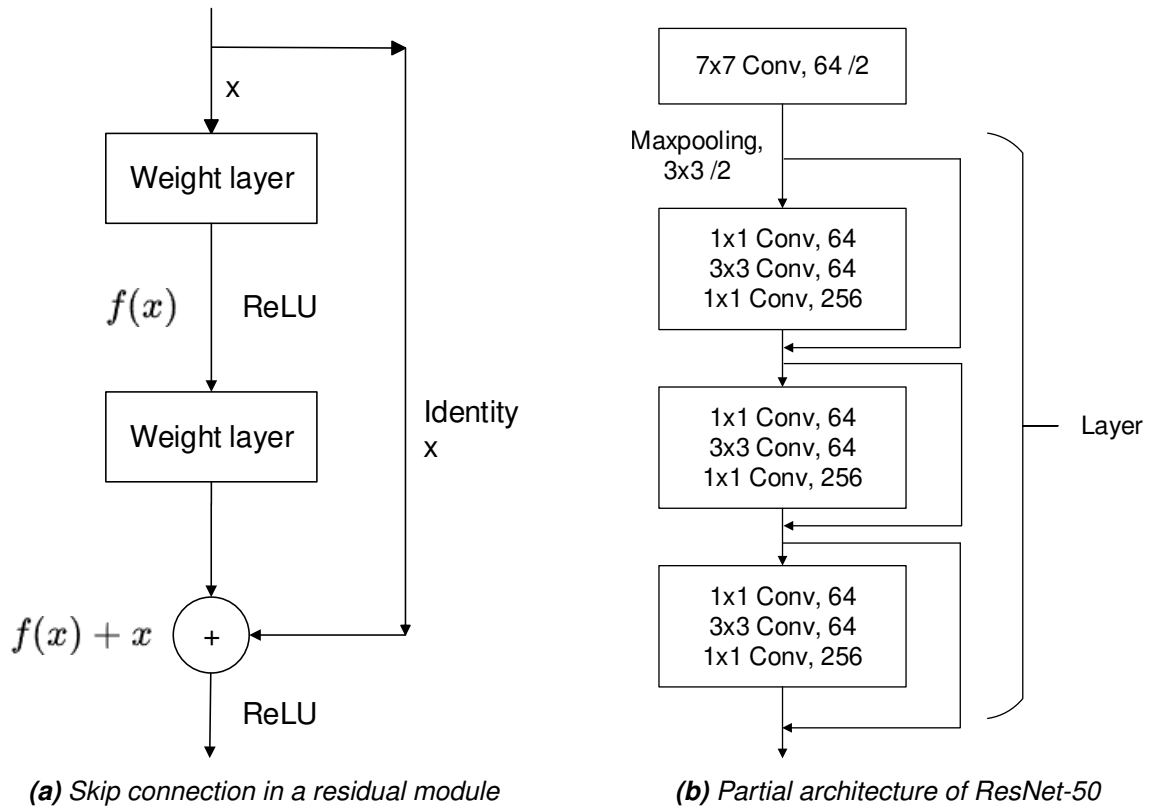


Figure 2.16. A residual module and partial architecture of ResNet-50, adapted from [9, p. 305-360]

2.5.1 Re-identification metrics

Typically, ReID models are evaluated using a Cumulative Match Characteristic (CMC) curve. The curve represents the re-identification rate at rank- k . Rank- k shows the probability that a correct re-identification is in the top k results in the list of result probabilities. Therefore, a rank-1 value at 100 % would indicate that every re-identification is done successfully, and the correct result is the one predicted as the most likely result. A rank-3 value at 100 % shows that the ground truth is always the result ranked within the top three results. The rank- k results for common datasets with the best model are shown in table 2.4. [47]

For a single-gallery-shot setting, which means that there is only one correct result for each gallery identity, the CMC curve is defined by averaging the top- k accuracy functions

$$Acc_k = \begin{cases} 1, & \text{if top-}k \text{ ranked gallery samples contain the query identity,} \\ 0, & \text{otherwise.} \end{cases} \quad (2.31)$$

which is a step function. For a multi-gallery-shot scenario, where each gallery identity may have multiple correct results, there is no standard definition for the CMC curve, therefore evaluation of different systems such as Market-1501 and CUHK03 is difficult. [48]. In

Table 2.4. Rank-k performances for datasets, adapted from [47]

Datasets	Best Combination	1	5	10
VIPeR	GOG-XQDA	41.1	71.1	82.1
GRID	IDE-ResNet-KISSME	26.6	43.1	50.9
3DPeS	IDE-ResNet-NFST _{exp}	53.4	77.8	85.6
CUHK01	GOG-NFST _{exp}	55.6	77.7	84.8
CUHK02	GOG-NFST _{exp}	57.9	79.3	85.7
CUHK03	GOG-kLFDA _{exp}	62.1	88.7	94.2
HDA	IDE-ResNet-NFST _ℓ	84.1	84.5	85.8
Market1501	IDE-ResNet-NFST _{exp}	64.3	80.9	86.2
Airport	IDE-ResNet-NFST _{exp}	42.7	67.5	76.0
DukeMTMC4ReID	IDE-ResNet-NFST _{exp}	54.6	68.6	73.7

in addition to CMC, also mAP can be used to evaluate a ReID system. As defined in equation 2.9, it calculates the average precision of the system for each re-identification and averages the results of all queries. mAP is more used in multi-gallery-shot settings.

2.5.2 Temporal Feature Aggregation

Temporal Feature Aggregation (TFA) is a technique used for object detection and re-identification tasks. TFA combines features from multiple viable frames to create a more robust feature representation. A form of TFA is simple averaging. It is defined by

$$F_t = \frac{1}{N} \sum_{i=1-N}^t F_i \quad (2.32)$$

where F_t is the aggregated feature at time t , f_i is the extracted feature from the i th frame and N is the number of past frames to count in the average. Another method is to use weighted averages of the features. A common method is to use the detection confidence as the weight. This provides that the uncertain results, that may be caused by occlusion or other visual problems, are considered less for the final feature vector. Weighted average is defined by

$$F_t = \frac{1}{N} \sum_{i=1-N}^t w_i * F_i \quad (2.33)$$

where w_i is the confidence from the i th frame. [49][50]

2.6 Hazard identification in computer vision

Hazard identification can be used for alerting workers or supervisors when the worker is entering a hazardous area. This area may be for example an area near a conveyor belt, a robot or a crane with the possibility of falling items. This can be achieved by marking a position in the camera viewfinder and alerting if a person is detected moving into the hazardous area. An example of this situation is shown in figure 2.17, where worker 2 is standing in the marked hazardous area and therefore should trigger an alarm. Worker 1 is not within the area.

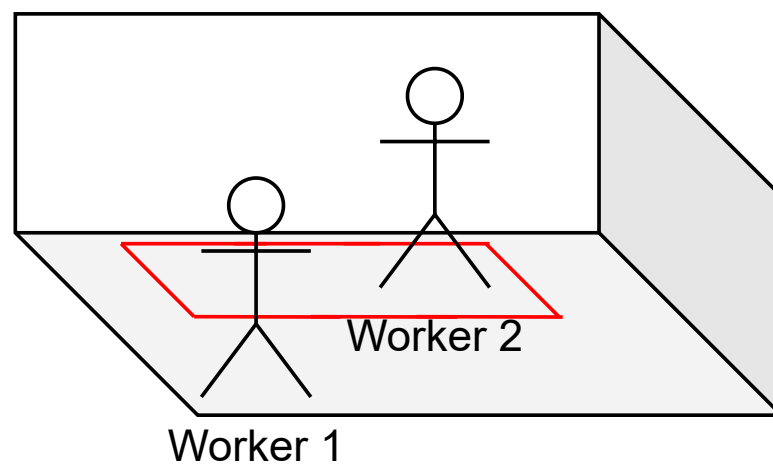


Figure 2.17. Marking a hazardous area

To find out whether the person is within the marked area in a 2-dimensional projection, the coordinates of the worker's legs need to be obtained. YOLO pose is a machine learning model that outputs coordinates of a person's body parts. The result that YOLO pose can achieve is illustrated in figure 2.18. By comparing the position of the worker's ankles with the marked hazardous area, an estimated position in the 3-dimensional space can be determined. The YOLO pose model is built on top of regular YOLO models. It is trained with a customized Microsoft Common Objects in Context (COCO) pose -dataset and it provides keypoint coordinates for pose estimation in addition to the bounding box that also regular YOLO models provide.

In keypoint estimation, Object Keypoint Similarity (OKS) is the most common evaluating metric. YOLO pose introduced an IoU based loss function for OKS instead of traditionally used a heatmap-based L1 loss, which has issues with the scale and type of the keypoint. The IoU approach allows for different penalizing depending on the keypoint. For example, a pixel-level error in keypoints such as the eyes is penalized more than a similar error in larger keypoints such as a shoulder because of the IoU, instead of only accounting for the distance between the prediction and the ground truth. The IoU loss variant used is Complete intersection-over-union (CIoU). The loss is defined as:



Figure 2.18. YOLO pose

$$\mathcal{L}_{box}(s, i, j, k) = (1 - \text{CloU}(\text{Box}_{gt}^{s,i,j,k}, \text{Box}_{pred}^{s,i,j,k})) \quad (2.34)$$

where (i, j) represent the anchor location and s the scale with the k^{th} anchor. [51]

Computer vision can also be used for visual relationship detection. In visual relationship detection, interactions between objects are detected. Examples of this in relation to a shop floor are “worker wear helmet” and “worker near robot”. A network developed for this task is a Visual Translation Embedding (VTransE) network. The network is capable of forming triplets in the form of “object 1, relation predicate, object 2” by utilizing computer vision, ontology, and natural language processing. [5]

2.6.1 Action detection

Action detection is used for detecting what a person is doing. Action detection is performed by a machine learning model called Spatial Temporal Graph Convolutional Networks (ST-GCN). The network is capable of detecting actions such as “walking” or “standing”. In addition to the actions, the model is also able to detect movement, such as “falling down” or “sitting down”.

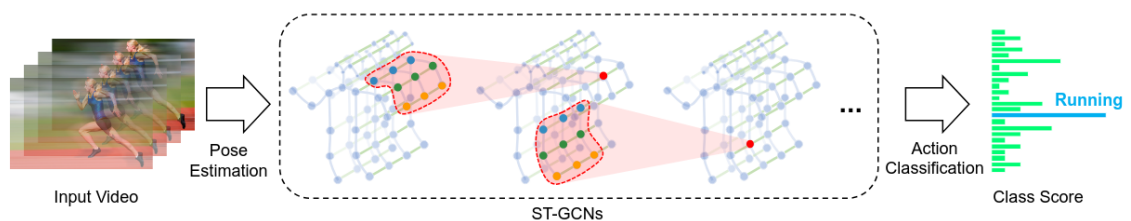


Figure 2.19. ST-GCN working principle [52]

ST-GCN works with 2- or 3-dimensional coordinates of human joints. While a traditional CNN has data in a grid, a ST-GCN has the data in a graph-based format, where joints are represented as nodes, connected by bones representing the edges. The model is able

to extract features from the skeleton using a spatial graph convolution and extract motion using temporal convolutions. [52]

3. DESIGN

This chapter outlines the design of the system. The system is designed to integrate real-time personnel tracking, hazard identification, and re-identification using state-of-the-art machine learning techniques.

First, the functional and non-functional requirements are presented. These requirements define the system's behavior and expected performance and other constraints. Then, the architecture of the system is presented, displaying how each component of the system interacts with the other components. UML diagrams are used to visually describe the system design and data flow. Finally, each module that the system consists of is described in terms of its purpose, behavior, inputs, outputs and integration with other components.

3.1 Functional and non-functional requirements

The system is defined by functional and non-functional requirements that are formed to describe the system's capabilities. These requirements ensure that the system meets its objectives for the given task.

3.1.1 Functional requirements

Functional requirements describe the functionality that the system should be able to perform. The requirements focus on the behavior and define the actions of the system. The functional requirements for the system are presented below:

Person identification

The system must be capable of identifying personnel in the camera frame. This includes both detecting a human and differentiating between individuals. The identification must work in non-optimal lighting conditions and handle up to 50 % partial occlusion of individuals.

Re-identification

The system must be able to ReID individuals that re-enter the frame after moving away from it. The system should maintain the individual ReID entry in the database for a period of 24 hours. This allows the system to keep the ReID database in order, as personal clothing may be subject to change after 24 hours.

Activity Recognition

The system must be able to determine the pose and actions of an individual. This includes detecting whether the person is lying prone on the shop floor or entering a restricted area, such as a robot working area. This requirement allows recognition of hazardous situations.

Event Triggering for Specific Scenarios

The system must be capable of triggering events based on preset conditions. For instance, if the activity of the individual is deemed hazardous, an alert should be shown in the system. These event triggers enhance the system's ability to mitigate risks in real-time.

Real-Time Performance

The system must operate in real-time to provide feedback and event triggers within a meaningful time frame. The video output should provide at least 5 frames per second in order to be considered reliable. This is essential for ensuring safety, as delays in detecting and reacting to hazardous situations could result in accidents or operational inefficiencies.

3.1.2 Non-functional requirements

Non-functional requirements set guidelines in which the system operates. These requirements describe the performance requirements and the scalability of the system. Non-functional requirements also focus on the external conditions in which the system should be able to keep its accuracy and performance. The non-functional requirements of the system are presented below:

Hardware independence

The solution must be compatible with generic cameras, ensuring that it is not tied to any specific camera vendor or proprietary hardware. The solution must support video input with standard full-HD resolution (1920 × 1080) and 30 frames per second. This

requirement enhances the system's adaptability and scalability across setups.

Scalability

The system must handle environments with up to five individuals in the frame while maintaining performance and reliability.

3.2 Architecture

The system has a main view that displays the output. The main view depends on four different modules that each provide for the output:

1. Object detection
2. Pose estimation
3. Personnel tracking
4. ReID

The object detector is responsible for detecting personnel in a given frame, while pose estimation detects the persons's extremities, hips and head position relative to the detected person frame, which is used for hazard identification. Person tracking makes sure that the re-identification is not done for every frame, and this allows the system to perform better considering the number of personnel visible in the frame, as re-identification is a resource heavy task.

The use cases of the system are shown in the use case diagram in figure 3.1. The system has a user as the only actor. The user is able to configure safety parameters by using a graphical user interface. The interface has options to draw a rectangle in the frame to determine the hazardous area to which the shop floor worker should not enter. The saved personnel data, that includes the latest saved ReID database entry. Each detected person has a bounding box marked on the camera preview window. The user interface also displays the number of personnel and has the main camera view with the system data overlaid to observe the shop floor.

The general flow of the system is described in the activity diagram shown in figure 3.2. As seen in the diagram, the first step is to capture a frame and pass it on to the object detector that detects all objects. The detector then filters the detections so that only people remain in the output that is passed on to the pose detector. The tracker then gets the data of the personnel, including the relative bounding box position and determines the track, and if necessary, the next step is to perform re-identification and save the person to the re-identification database.

The activity diagram in figure 3.2 also shows the triggering of alarms. As seen in the

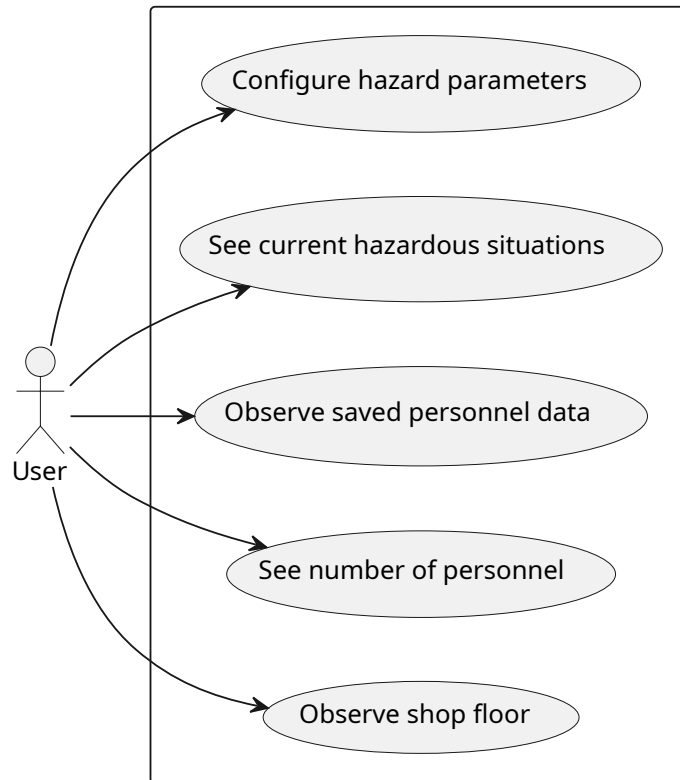


Figure 3.1. Use case diagram

diagram, the alarms should be triggered when either the individual enters a restricted area or the individual appears to be in a personal emergency. The restricted areas are marked by the system operator on the graphical user interface. The personal emergency is assumed to be ongoing if the individual is lying on the floor for a prolonged period of time.

The classes that the system contains are described in the class diagram that is presented in figure 3.3. The system directly has four classes that it manages. `VideoCapture` is an asynchronous class that reads the video feed from a source. It has a public method for getting the latest frame, and public methods starting and stopping the system. The `ReID` class handles the re-identification task. It has methods for getting a person identification based on an image of the person or pre-extracted features, and a method for extracting features given an image. It also has a method to update the `ReID` database with a given image or extracted features. In addition, the `ReID` class provides methods for activity recognition given a list of the individual's keypoint from the last thirty frames. The `Tracker` class contains the logic for MOT, which is handled by the `Update`-method.

The `Pose` class handles both the pose detection and object detection. First the class receives the frame, runs the inference on it, identifies the pose, and outputs a vector of `PoseDetection` elements, that contain the positions of each detected body part.

The system maintains two lists or databases: `tracked_persons` for person tracking,

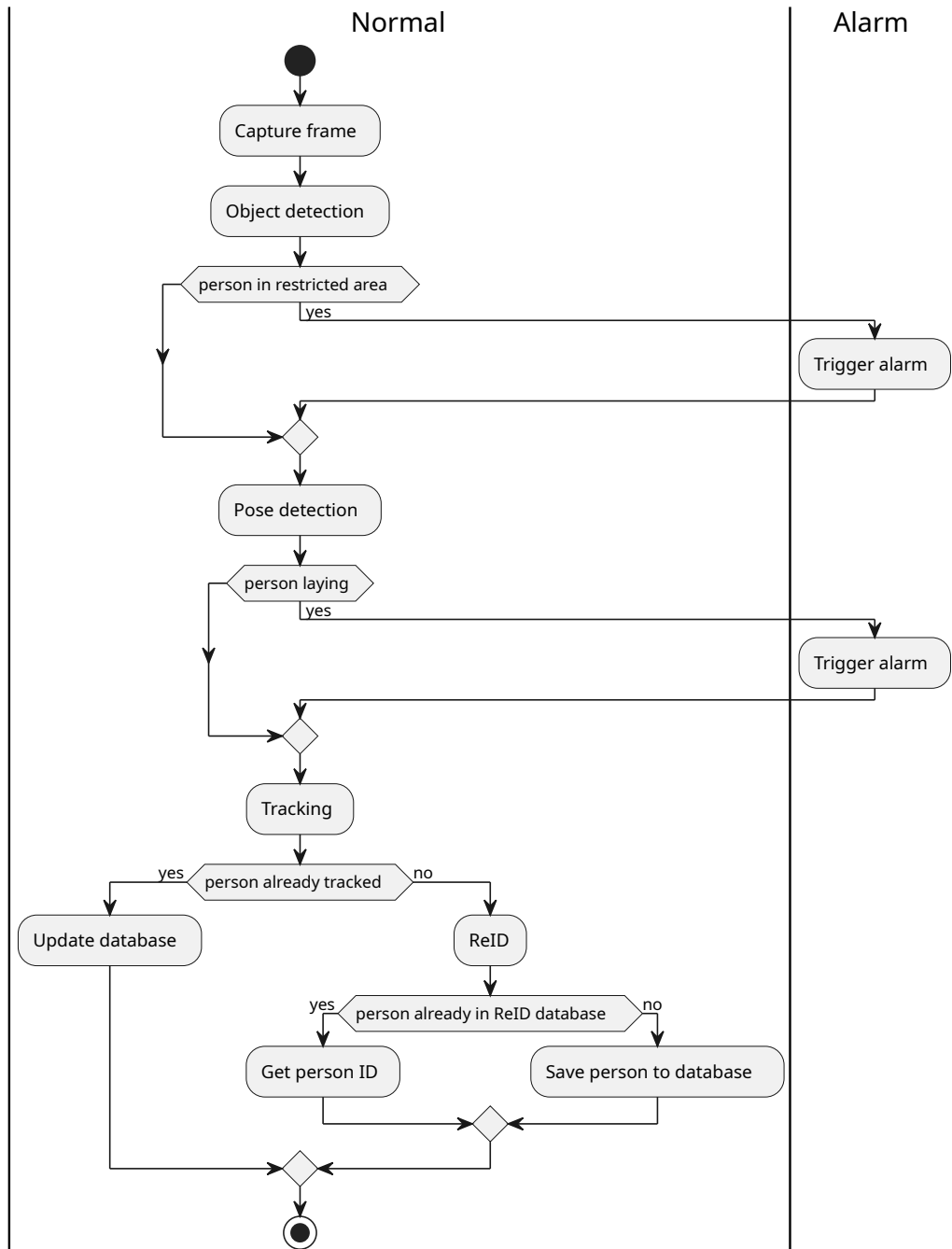


Figure 3.2. Activity diagram

and `identified_persons` for re-identification. Both of the lists are made of items with type `TrackedPerson`. This class holds an ID, detected bounding box area in pixels, and reference to the picture of the detected person.

The sequence diagram in figure 3.4 shows the connections and data flow between the modules and databases of the system. As shown in the diagram, the main program receives data to be displayed from the video stream as a frame, the detector as the bounding boxes, the pose identifier as keypoint data and finally the personal identifier that ultimately comes from either the ReID database or the MOT track database.

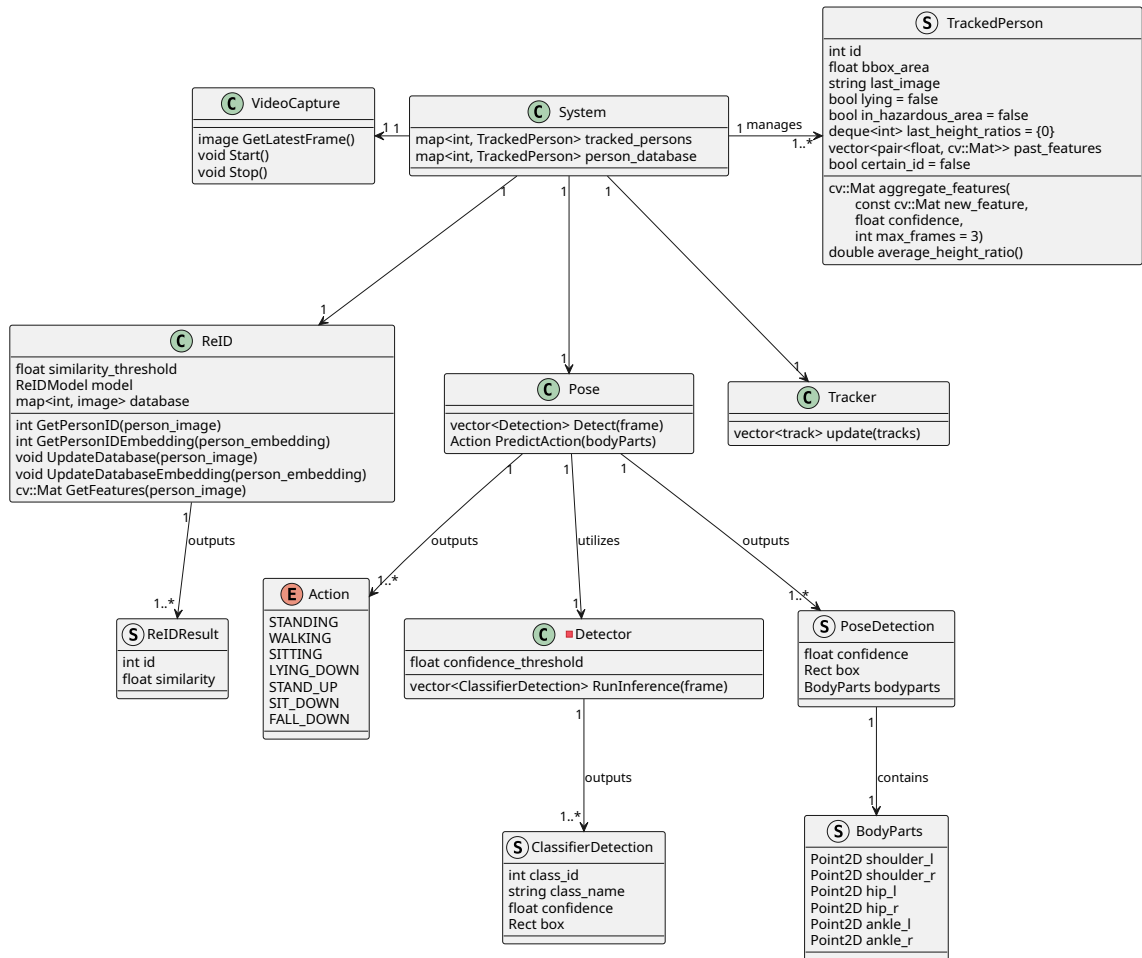


Figure 3.3. Class diagram

3.3 Object detection

Object detection forms the foundation of the system. Its role is to detect personnel from a video feed and pass the detection data further in the process. The following modules all depend on the output of object detection. Robust detection is needed for the system to work in real-time, and in non-optimal lighting conditions. Some personnel might also be partially occluded, and there may be multiple people within the scene.

For this task, there are multiple common state-of-the-art choices, such as YOLO, SSD and Region Based Convolutional Neural Network (R-CNN). However, YOLO is able to top the alternatives when it comes to performance. Another benefit of YOLO is the available model sizes. The YOLO model can be chosen from multiple options ranging from nano to extra-large. This allows the model size to be chosen based on available hardware, however the smaller models have less optimal mAP performance values. Finally, in addition to object detection, a YOLO pose estimation model is available, which increases the value of YOLO for this system. Pretrained models are also provided by YOLO developers that have been found to be functional in this setting. The object detection and pose estimation models have been trained with the COCO and COCO-pose datasets respectively.

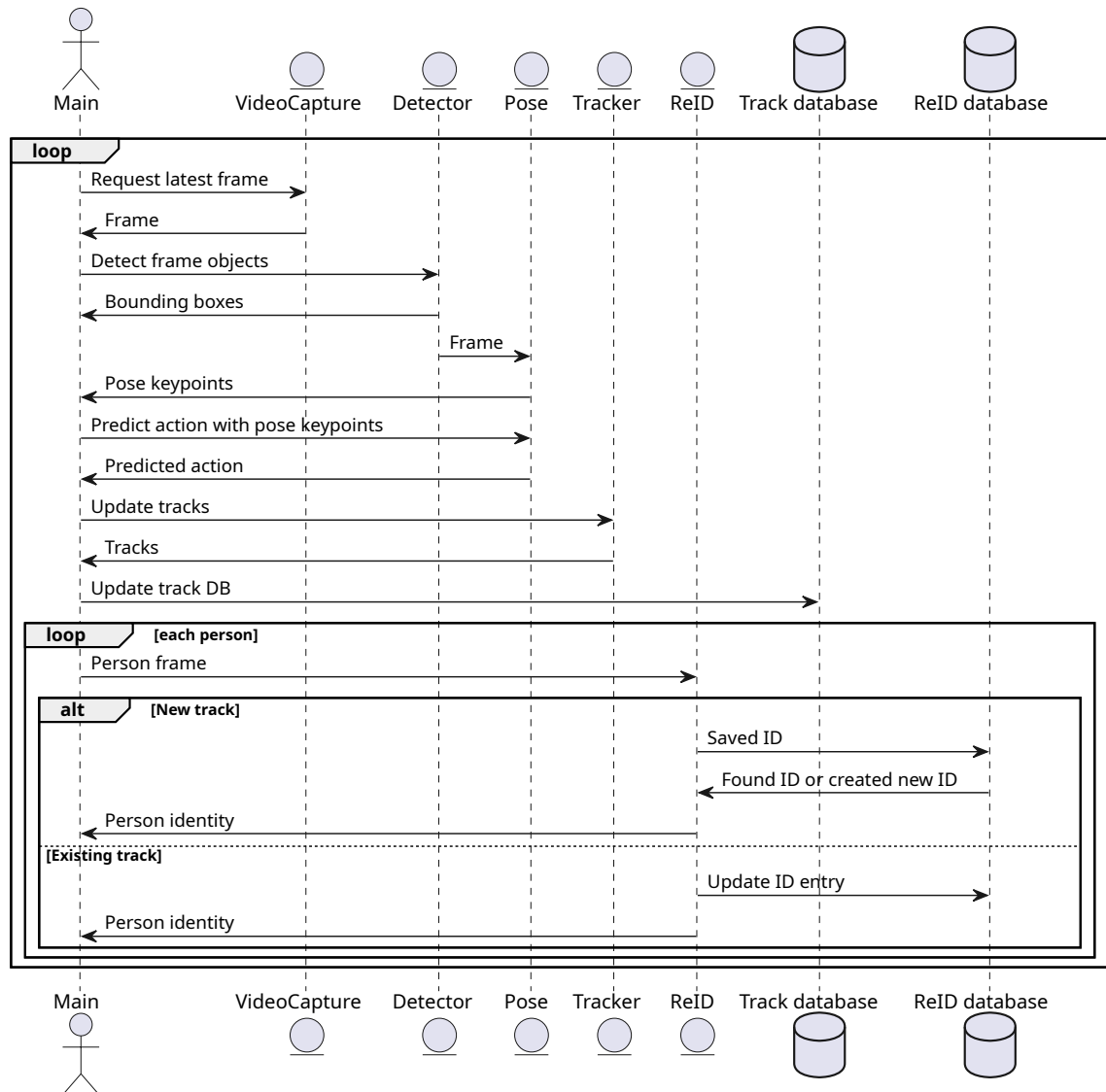


Figure 3.4. Sequence diagram

The data used for object detection comes from the video stream. The detector grabs the latest available frame that is available after the previous detection process is ready to avoid delay in the stream. The detector can work with unmodified images and no manual preprocessing is required outside scaling of the image, but in bad lighting conditions more preprocessing may be necessary.

The workflow for object detection has three steps:

1. Grabbing the latest frame.
2. Performing detection.
3. Outputting bounding boxes and confidence scores.

This data flow is illustrated in figure 3.4.

Challenges regarding object detection are only partially visible individuals, false positive

handling and accuracy in poor lighting conditions. The occlusion problem is handled by the detection algorithm YOLO with its advanced features such as local features. False positive handling can be optimized by finding appropriate threshold values for detection confidence, or it can be mitigated using more powerful hardware and switching to a larger model. Some false positives can also be ignored, as they do not necessarily provide negative input for the system. The detector can be helped during poor lighting conditions by preprocessing the images to provide more contrast and improve the image quality.

3.4 Pose estimation

Once personnel are detected, pose estimation uses the data from the detector for analyzing body part positions. Pose estimation is the component in the system that is responsible for generating output depicting the keypoints of the human body for each of the individuals in the frame. The same challenges as in object detection also apply for pose estimation, with the more significant one being occlusion, as the pose estimation would have to guess the positions of the keypoints when they are occluded.

For this module, YOLO has a pose estimation model. Like the object detection model, pose estimation also has different sized models for various use cases and hardware. The model has pretrained versions that have been trained with the COCO-pose dataset. The pose estimation model works directly with the object detector. It handles the detection, filtering and pose estimation itself, without the need for a separate object detector in addition to the pose estimator.

The YOLO pose estimation model takes an image as input, and it outputs the keypoints for each body part. It is able to determine the position of the head, shoulders, hands, hips, knees and ankles with mAP-50 between 81,0 % and 91,1 % depending on the size of the model used. The speed ranges from 52,4 ms in the smallest model to 488,0 ms on the largest when running on a CPU, which would translate to about 20 frames per second in the best-case scenario.

3.5 Action recognition

The action recognition module is responsible for detecting the actions of the individuals in the frame, mainly related to hazard identification. The module is able to detect actions such as walking, standing, sitting, lying down, and falling down. The module is able to detect these actions using a ST-GCN model. The model requires 2-dimensional coordinates of human joints. The network is able to extract features from the skeleton using spatial graph convolution and extract motion using temporal convolutions.

The workflow of personnel tracking has two steps:

1. Collecting keypoint data for N frames for each person.
2. Forwarding the data to the model.

This data flow is illustrated in figure 3.4.

Main challenges regarding action recognition are occlusion and accuracy. If the joints are found in incorrect positions, that will directly affect the performance of activity recognition.

3.6 Personnel tracking

Personnel tracking is a main key component in the system. Personnel tracking helps in keeping the continuity of the identities for the individuals without the need to perform resource-heavy ReID task on each detected person. Other objectives are handling occlusion based on movement, where a traditional detector would have trouble. It is also able to identify reappearing individuals.

The workflow of personnel tracking has three steps:

1. Receiving detection data.
2. Associating individuals across frames.
3. Performing short-term re-identification.

This data flow is illustrated in figure 3.4.

Main principles guiding the design of personnel tracking is accuracy, robustness and real-time performance. Optimizing accuracy ensures minimal tracking loss and minimizes identity switches and robustness handles changing environment illumination. The tracking algorithm also needs to support tracking multiple individuals in a single frame and should be able to work with the output from the detector module of the system. It is also important that the tracker does not use resource heavy deep learning algorithms to keep the system performance real-time.

The algorithm chosen for this task is ByteTrack, which adopts a Kalman filter in order to track the bounding boxes [41]. The algorithm was chosen based on the accuracy metrics and performance. Also, ByteTrack is able to detect occluded or low-confidence objects that traditional MOT algorithms do not consider relevant and would dismiss [41].

The main challenges in personnel tracking are occlusion and identity switches. ByteTrack handles occlusions by considering low-confidence detections, ensuring better tracking continuity even in challenging conditions. This means that the tracking algorithm will calculate position changes also for objects that the detector did not find very likely to be of the target type, which can happen if the person is either in bad lighting, occluded, or if the camera view is blurry. Identity switches can happen when people pass each other, and the algorithm is unable to keep track of the identities. ByteTrack handles this with

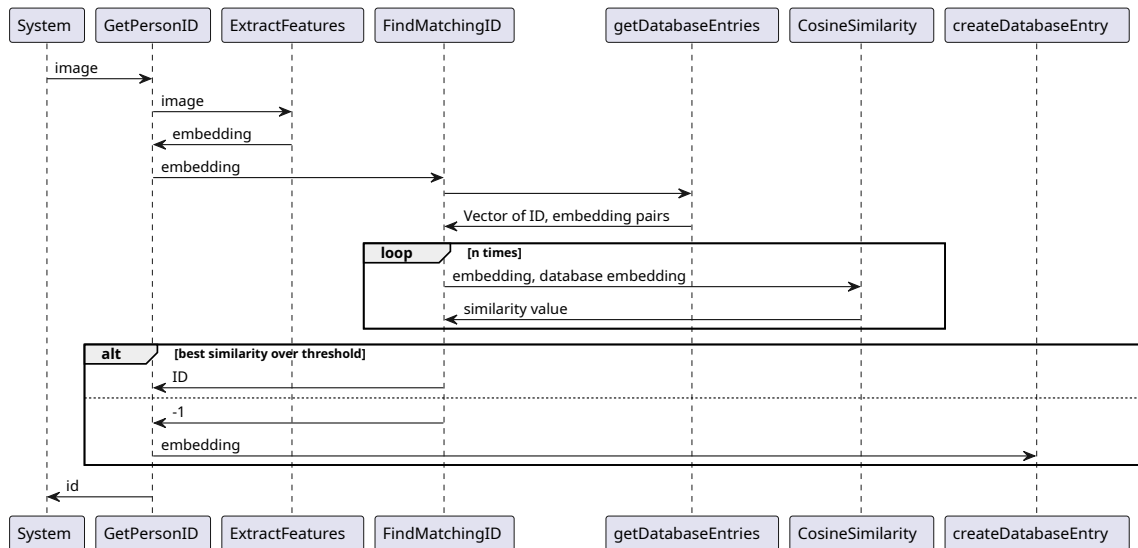


Figure 3.5. *ReID sequence diagram*

Kalman filter and given that the speed of the individuals remains predictable, the success rate is expected to be high. However, a deep learning algorithm could perform better if there were a large number of personnel, but it would be at the cost of performance.

3.7 Real-time re-identification

Real-time re-identification is important in maintaining the continuity of individual identities in the long term. The feature is used in scenarios where there must be more than one person working at a shop floor to ensure safety, but it is possible that all personnel are not visible in the camera view at one time. If the system was set up such that the two individuals were not identified as separate identities, it would not be possible to estimate the number of individuals on the shop floor.

The workflow of re-identification has three steps:

1. Receiving detection data and extracting features for each individual that were not present in the previous frame.
2. Associating individuals with existing feature extractions in database.
3. Identity assignment or giving a new identity if a match is not found with a given threshold.

This data flow is illustrated in figure 3.5.

The deep learning algorithm chosen for the re-identification task is ResNet-50. ResNet-50 is able to extract the features from a bounding box of a person. The extracted data can then be saved into a database in vector format identified by the person id, and it can be compared to later using a cosine similarity function with a given threshold. For real-world use cases, there needs to be a set time period after which the database entries

are dropped, as personal appearance typically changes by the day. For this system the time limit will be twenty-four hours. Main challenge within the twenty-four-hour period is changing natural lighting. The features of the extracted frame may be changing based on how the lighting changes. The database entries of each individual should be updated periodically in order to mitigate the issue, if the person is in the frame for an extended period.

Re-identification also utilizes TFA. In the process of saving the features of a newly discovered individual, the system takes three samples of the person with one second intervals. A weighted average is then calculated from the three-feature vector weighted by the object detection confidence. This is done to ensure that the system does not only save an occluded image of the individual but instead increases the possibility of high-quality images. While performing the ReID of an existing track, the system also takes three samples of the person with one second intervals, and the weighted average is calculated in the same way. The weighted average is then compared to the database entries with a cosine similarity function.

4. IMPLEMENTATION

This chapter implements the empirical part of the thesis. The purpose is to implement a system that tracks personnel in shop floor and is able to trigger alarms based on user-given conditions, such as an individual accessing a restricted area, or other criteria based on the relative locations of the individual's extremities. First, the implementation of the modules is introduced. Finally, the environment that the system is placed in is introduced.

4.1 Video capture

The first module in the system pipeline is the video capture system. Its sole task is to read input from a camera device attached to the computer and output the latest frame when requested. For testing purposes, the module also supports reading a given video file. The video capture module runs asynchronously and utilizes the OpenCV library. This allows for reduced delay and separation of the video capture from the main system loop, allowing for better performance, as only the latest available frame is read. OpenCV is able to read a video stream with the given functional requirements, full-HD resolution (1920×1080) and 30 frames per second.

OpenCV stands for Open Computer Vision Library. It provides tools and functions for real-time computer vision applications. In the latest version, OpenCV provides a C++ Application Programming Interface (API). The components of OpenCV are widely used in the project. The most notable being a `cv::Mat` data type that represents an n-dimensional dense array class. This class holds for example all the image and extracted feature data.

The video is captured from the external camera using a `cv::VideoCapture` object. The method `cv::VideoCapture::read(OutputArray image)` runs in a continuous loop, and provides the read frame to the given memory address. The loop is set to sleep for the duration given by calculating frame time from the FPS value of the stream.

4.2 Implementation of object detection

The object detection module has a task of identifying personnel in the frames that it gets from the video capture module. This module is the foundation of the following modules, being important in providing data to MOT and ReID modules.

For the implementation of object detection, the pretrained YOLO11 model can be obtained from the project's GitHub page. The model is provided in a PyTorch Tensor format that needs to be converted to Open Neural Network Exchange (ONNX) format for use with OpenCV using the C++ API. The converted ONNX model can then be loaded as a `cv::dnn::Net` object with `cv::dnn::readNetFromONNX`.

The object detection is performed in the following steps:

1. Target image is resized to a square.
2. The image is preprocessed with OpenCV method `cv::dnn::blobFromImage` that normalizes the pixel values.
3. The image is set as the input (`cv::dnn::Net::setInput`) for the loaded ONNX network and a forward pass is run on the network (`cv::dnn::Net::forward`).

This process returns output data in a vector. The output vector has one element with dimension $1 \times 84 \times 8400$, where 1 is the batch size, typically only one image. The 84 values represent the following data:

- Center x: x-coordinate of the bounding box center.
- Center y: y-coordinate of the bounding box center.
- Width: width of the bounding box.
- Height: height of the bounding box.
- Confidence: indicating the likelihood of an object being present in the bounding box.
- Class scores: probabilities for each of the 80 possible classes.

The 8400 is the amount of anchor points to which the image has been split to, and some of the points contain a detected object. In order to convert the vector to more easily readable values, the vector needs to be reshaped to a 2-dimensional matrix where each row represents a detection. The matrix is then transposed. Now the matrix has dimension 8400×84 , with each row containing the same data as listed above for all 8400 anchor points.

To access the data, the `cv::Mat` provides an attribute `uchar *cv::Mat::data` that is a pointer to the start of the data. The vector is then iterated over the count of rows and the data is found from the given pointer by appending the size of one row on each iteration. Then, to extract the detections found from the 8 400 total results, the matrix is filtered for the detections where any class probability is greater than a threshold `model_score_threshold_` of 0,45. Finally, Non-maximum suppression (NMS) is performed with the OpenCV method `cv::dnn::NMSBoxes`. NMS is needed to filter duplicate results, as typically the anchoring technique used by YOLO returns duplicates.

4.3 Implementation of person tracking

Person tracking is implemented with a MOT algorithm ByteTrack. ByteTrack utilizes a Kalman filter to associate bounding boxes in the current frame with bounding boxes from the previous frame in order to maintain identities during the video stream. Its advantage is that both high-confidence and low-confidence detections are associated across frames with the Kalman filter, which ensures improved robustness. ByteTrack maintains vectors of the currently active tracks, lost tracks and removed tracks, which helps in maintaining accuracy. The lost tracks can be re-enabled as active in case the detector has lost sight of an individual for a short period of time due to occlusion or error.

The tracker is activated by the `BYTETracker::update` function, that takes a vector of ByteTrack objects as a parameter. The object contains information of the detection bounding box rectangle, a class label and a probability. Then the algorithm associates the detection with existing tracks or creates new tracks if necessary and categorizes the tracks into the tracked, lost, and removed vectors. The tracker returns data as a vector of `Strack` objects. These objects contain information on the track identification.

The returned track data is then iterated over each individual. The main application maintains a vector of tracked people and each ByteTrack result is verified whether it is already a tracked person, or a newly appeared person. If the track belongs to an existing identity, the track identification is already present in the maintained vector of tracked people. Otherwise, the ReID algorithm tries to find the identity of the track.

4.4 Implementation of re-identification

Person re-identification ensures that personnel leaving and entering the camera's view keep a stable identification, even after being gone from the view for a longer period of time. This allows for continuity and better tracking of the shop floor. For example, if an individual moves away from the camera's view but remains in the shop floor, the system should be able to know that upon re-entry, the identities have not changed, and the total number of people in the shop floor remains same. Without re-identification the situation described could end up in the system assuming that a new person has entered the space.

Backend of person re-identification is implemented using ResNet-50. ResNet-50 is responsible for extracting data from an image that is cropped to contain mostly the person in question. The feature extracting model is modified from TorchVision-provided ResNet-50 model. The final layers of the model are modified, leaving the extracted features as the final layer instead of a fully connected layer that would provide the class probabilities. Finally, a custom embedding layer is added to reduce feature dimension and a batch normalization for stability. The algorithm returns data in a vector of floating-point numbers.

The system maintains a database that saves the identification with the extracted features of the people as a pair. In addition to new identities that are added to the database on discovery, the existing identities are updated. The system sets rules for updating the entries. First, the system updates the embedding saved in the database using the weighted feature extraction to ensure up-to-date database entries. Another rule is that the entries are discarded after 24 hours to avoid growing the database with unused data. It is assumed that in the context of this system, the identities are irrelevant after a workday, or 24 hours after the last discovery of an individual.

The ReID module is started by the system that requests for a person's identification based on an image of that person. The `GetPersonID` function takes the image as a parameter and passes that image for ResNet-50 for feature extraction. The extracted features are further passed for `FindMatchingID` function, that loops over each database entry comparing the system-given features with each of the database entries using a cosine similarity function. Cosine similarity is defined as:

$$\frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \times \|\vec{b}\|} \quad (4.1)$$

Where \vec{a} is the system given features and \vec{b} the database entry features. The best match is then compared to a threshold of 0,92. If the best similarity exceeds the threshold, the module assumes a match is found. Otherwise, it will return a newly created identity back to the caller.

TFA is implemented in the ReID system by storing three one second apart images of the person in its data structure. The system then calculates the object-detection confidence weighted average of the features and provides the ReID result after the three iterations. After the three features are received, the identity is compared against exiting identities and saved in the database if necessary. This is done to ensure that the system has a stable identification of the person before providing the identification to the main system. While slowing down the ReID process for three seconds, the identification of a person is not considered critical information in terms of real-time performance.

4.4.1 Re-identification model training

ResNet was originally created for image recognition, and therefore it needs to be trained with a ReID suitable dataset. Market-1501 is a dataset containing images of people captured with six different cameras placed in different locations. Of the six cameras, five are high definition (1280×1080) cameras and one is a standard definition camera with a resolution of 720×576 . The different camera positions and resolutions guarantee that each person is captured with slightly different perspectives and levels of detail, which in

turn is useful for the training dataset. The dataset contains 32 668 bounding boxes of 1 501 identities and each identity is captured by at least two of the cameras to ensure cross-camera search. [46]

The ResNet-50 model is trained with the Market-1501 dataset using Python. The training parameters are:

- Batch size: 32
- Learning rate: 0,05
- Number of epochs: 60
- Stride: 2

Using these parameters, the model is able to achieve Rank-1 of 88,84 % and mAP of 71,59 % when testing the model with evaluation data.

4.5 Implementation of hazard identification

Hazard identification is implemented using YOLO pose estimation. The model returns keypoint data for each person in the given frame. Keypoints include ankles, hips, shoulders, hands and head. Utilizing these keypoints, the hazard identification algorithm can be implemented. The model used is a pretrained model provided by the YOLO11 developers.

The graphical user interface contains the ability to draw a rectangle for the positions that the system should consider hazardous. This area can be for example the area around a robot. The system will then check whether any person on the shop floor has the keypoints for their legs within this area. Given that the marked area will often expand to behind the robot, the system will need to estimate the position of the keypoints in some circumstances.

Another hazard the system is able to identify is a person lying on the shop floor. In a three-dimensional space, the checking for this position is difficult using only 2-dimensional coordinates of the person's joints. If the person is lying on the floor along the horizontal axis in terms of the camera view, the system can easily note that a line drawn from shoulder to ankle through the hip is nearly straight. However, if the position happens in the vertical axis in terms of the camera view, the system needs to determine whether the person's dimensions have changed relative to the time when they were standing up. The situation is illustrated in figure 4.1. Given the positional challenges, for detecting a lying person, a ST-GCN model is used. It is able to detect the following positions: standing, walking, sitting and lying down. In addition to the current positions, the model is also able to identify the following actions: stand up, sit down and fall down. The model takes 30 frames of keypoints data provided by YOLO11-pose that is normalized and fed

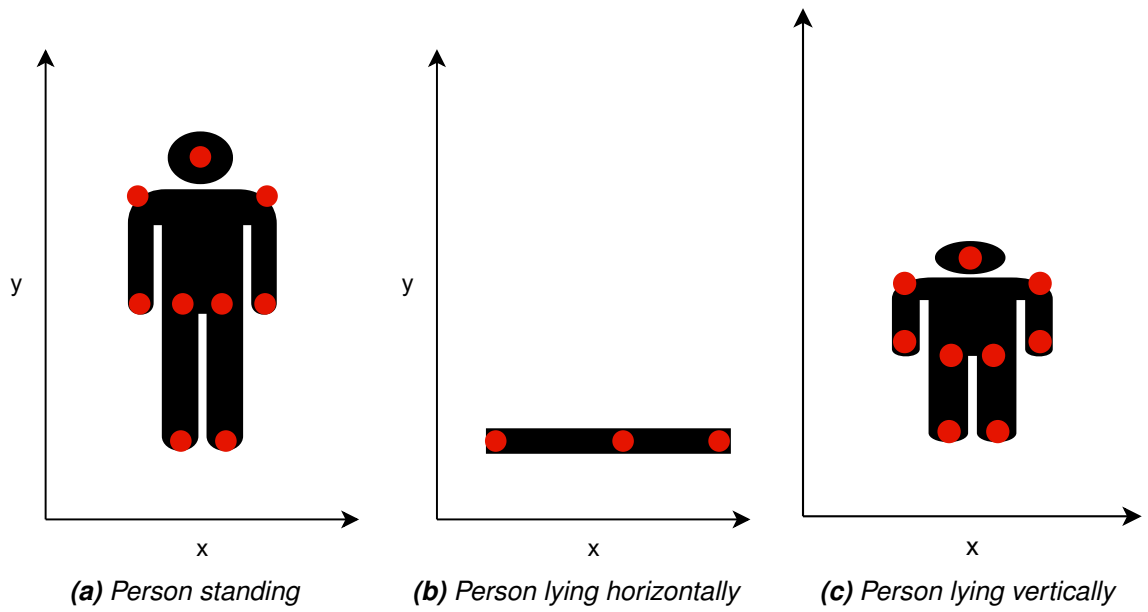


Figure 4.1. Estimating positions

to the model. This causes the model to take roughly one second to process an action, depending on the camera frame rate. The model then returns the probabilities of each action that the person is performing. The model is trained with the Le2i dataset. [52]

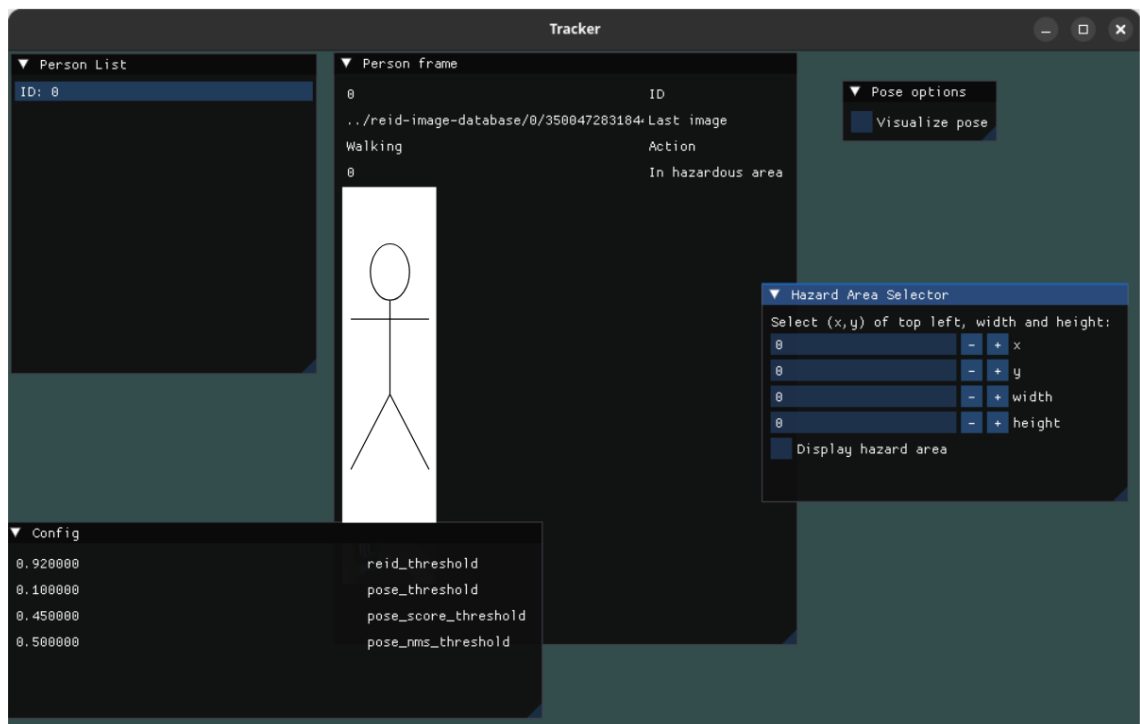


Figure 4.2. Graphical user interface

4.6 Implementation of the graphical user interface

The graphical user interface is implemented using the Dear ImGui library. The library is chosen for its ease of use and the minimalistic design patterns. The library has only minimal dependencies, allowing easier portability between systems. The application is divided into two windows: the video feed, the control panel. The video feed displays the video stream from the camera. The control is illustrated in figure 4.2.

The control panel contains the controls for the system. The user can set the parameters for the system, including the model score threshold for the object detection, the ReID threshold for the person re-identification, the hazard area rectangle, and the angle of the floor relative to the camera perspective. The interface also contains options to visualize the pose of a person, lists the personnel on the shop floor, and the alarms that have been triggered. For each person in the shop floor, the video feed visualizes the identification number, the bounding box, and the pose keypoints. Selecting a person in the control panel allows the user to see the saved data of the person. This data contains lying hazard identification status, positional hazard identification status and the identification number.

5. RESULTS

This chapter presents the evaluation results on the created system. The performance is measured by multiple metrics, including real-time efficiency, re-identification reliability, false positives regarding hazard recognition and overall system robustness. This chapter also discusses the effectiveness of the implemented system in handling industrial environments with dynamic conditions such as occlusions.

5.1 Experiment setup

A test video was created in a FastLab. The laboratory environment is shown in figure 5.1. As seen in the photo, the testing environment has typical shop floor appearance, with various objects and colors and items that cause occlusion. The test video was recorded at 30 FPS with a resolution of 1920×1080 pixels. The test video covered the following test cases:

1. Re-identification: A person enters the frame, walks around, and exits the frame. The person re-enters the frame after a few seconds. A total of three people entered the video on numerous occasions.
2. Occlusion: A person enters the frame, walks behind an obstruction, and exits the frame.
3. Pose hazard identification: A person enters the frame, lies down on the floor.
4. Positional hazard identification. A person walks to a restricted area.

The system is fed the recorded video, and the tests were performed on a computer using a CPU, AMD Ryzen 7 3700X.

5.2 Personnel detection & tracking accuracy

The object detector had no issues detecting personnel from the given image sequence. The detector had no false positives nor false negatives, therefore the precision and recall values are both at 100 %. Test subjects kept their track identities generally through the testing. However, due to the low system performance, the FPS had an effect on tracking performance. Given that the system was able to run at 5 - 10 FPS depending on the

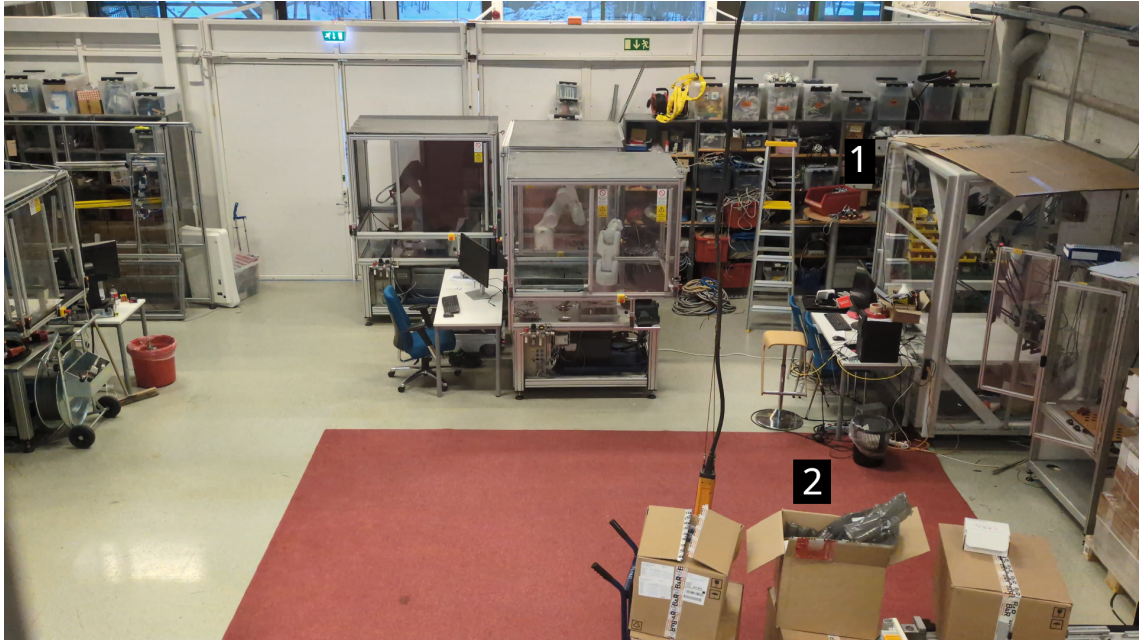


Figure 5.1. FastLab

number of visible individuals, the tracking algorithm had issues when the movement of an individual was not predictable. The cases where the system was losing track identities were mainly when the subject walked behind a hanging wire, marked with number 2 in the figure 5.1. The tracking algorithm achieved the following MOTA:

$$\text{MOTA} = 1 - \frac{0 + 0 + 4}{10} = \frac{6}{10} \quad (5.1)$$

The MOTA value is therefore 60 %. However, a lower value does not lower the overall system performance significantly. The effect of the tracking algorithm performance on the system is mainly related to the frequency of ReID cases.

5.3 Re-identification accuracy

Testing of ReID was conducted by having the same people enter the frame multiple times. The system was evaluated based on how many correct re-identifications the system was able to perform. The ReID module was able to re-identify the person correctly with an accuracy of 80 %.

The system failed to re-identify one person out of five total ReID cases. The person was previously assigned an identification number of 3, but the system re-identified the person as an entirely new person. The probability of the person being the original identification was found to be only 83 %. This low probability is likely caused by occlusion in the saved frame. While the re-identification system worked well in most cases, the system could be improved by using a larger object detection model. It was noticed during the testing

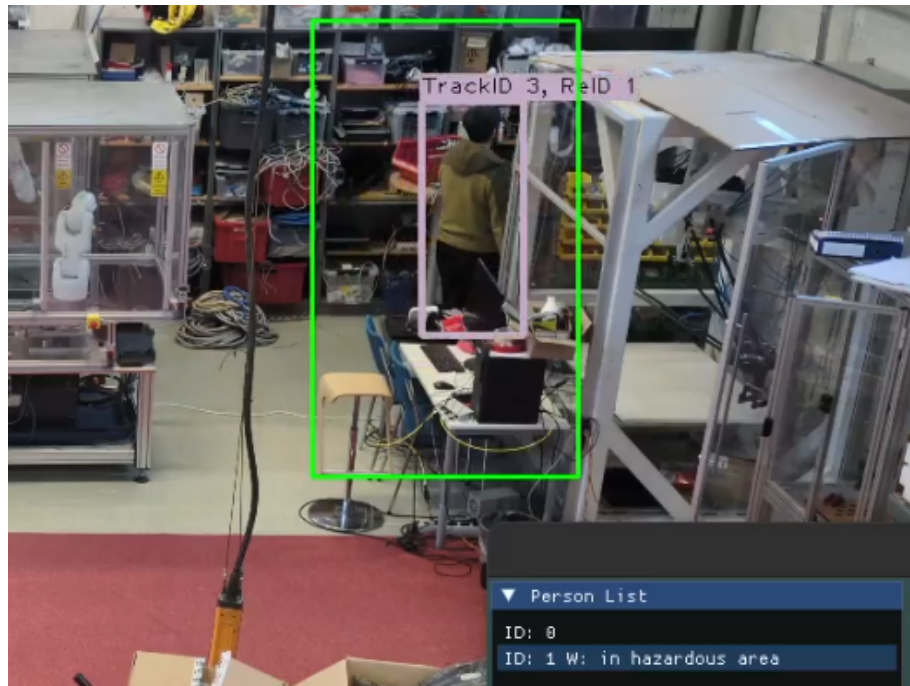


Figure 5.2. Test of positional hazard identification

that the system often identifies a larger bounding box than necessary, which causes there to be more background in the image than what would be optimal for the re-identification system.

5.4 Hazard identification

For testing the hazard identification module, a zone was marked as hazardous and a person walked into that area. The results are shown in figure 5.2. As seen in the figure, a person in the hazardous area is shown in the graphical user interface. The system was able to determine the position of the personnel by their legs and therefore identify whether the person has entered the marked area.

The system was also tested for pose-based hazard identification. The test subject lies on the floor two times. The system was able to detect the pose of the person with an accuracy of 50 %. However, separate testing of the pose-based hazard identification showed that the activity recognition model works significantly better when the FPS is at a higher level. This is caused by the fact that the system compares the last thirty frames of individual poses to determine the action. Therefore at 10 FPS, the system sees the last three seconds, whereas a 30 FPS system would consider one second.

5.5 Limitations

During the testing of the system, a few limitations were identified regarding both the test system and the developed system. Within the developed system, the performance could have been significantly better had the system contained a Compute Unified Device Architecture (CUDA) -enabled Graphics Processing Unit (GPU). Regarding the developed system, the activity recognition module has issues within the test scenario; it does not consistently detect a lying person if the person does not rapidly fall down, therefore placing oneself in a lying position in a controlled manner did not always trigger the required output. However, the system was tested separately for the functionality and was found consistently functional, when the falling happens in a realistic manner. Also, increasing the amount of camera angles would significantly improve the tracking performance of the system, which can be limited by occlusion.

6. CONCLUSIONS

The aim of this thesis was to create a system that is able to provide improvements for hazardous situations in industrial environments. The system was developed to detect personnel, track their movements, and identify hazardous situations. The system was tested in a realistic shop floor environment, and the results showed that the system was able to detect personnel, track their movements, and identify hazardous situations successfully.

6.1 Research questions

This thesis was able to answer the research questions presented below:

1. How to optimize personnel tracking in real-time dynamic environments?
2. How to maintain the reliability of re-identification?
3. How to implement identification of hazardous situations?

The conclusion for the first research question was found to be an effective tracking algorithm that is able to track personnel in real-time. The algorithm used in the system is ByteTrack, that is an object tracking algorithm that utilizes a Kalman filter and Hungarian algorithm to track objects. ByteTrack is also found effective in variable lighting situations and spaces that cause occlusion given that it is designed also track the less-likely probabilities, which should reduce identity switches and track losses [41]. Personnel tracking was evaluated using the MOTA metric, which showed that the system was able to track personnel with a MOTA value of 60 %.

The second research question was answered by using a re-identification model that is able to re-identify personnel in real-time. The ResNet-50 model was found to be adequate for re-identification, even more so after being trained with a re-identification dataset. The re-identification module was evaluated by comparing the re-identified personnel with the ground truth, which showed that the re-identification module was able to re-identify personnel with an 80 % accuracy.

The last research question was answered by using a hazardous situation detection algorithm that is able to detect hazardous situations in real-time. The algorithm has two parts: the first part is an action detection model, that is able to detect a person falling.

The second part detects if a person is moving to a hazardous area. The area is marked in the system by a user. The action detection model was evaluated by comparing the detected actions with the ground truth, which showed that the action detection model was able to detect actions with 50 % accuracy. The hazardous area detection was evaluated by comparing the detected hazardous situations with the ground truth, which showed that the model was able to detect hazardous areas with a 100 % accuracy.

6.2 Future possibilities

As machine learning technologies continue to evolve, there are an increasing number of possibilities for improving the developed system. Also, as more powerful hardware becomes available, heavier and newly developed models may be used, which could improve the system performance. In general, the system would benefit from using a multi-camera setup to detect hazardous situations more accurately and collect more reliable data.

As the YOLO model is under continuous development, it is likely that future versions will have even greater accuracy. Currently, the largest YOLO11 models achieve mAP 50–95 value of 54,7. While there are models with a better precision, YOLO is known for its speed and therefore suitability for a real-time application. The issue with the current version YOLO related to the needs of the developed system is the bounding box of the detection, that is initially estimated to be too large.

Re-identification is a challenging task, especially in dynamic environments. The object detection accuracy is critical for improving ReID. The system relies heavily on the detected bounding box, and if the box is not accurate, the re-identification performance will suffer. A multi-camera setup could improve the re-identification performance by providing more data for the re-identification model.

REFERENCES

- [1] Hao Wu and Jinsong Zhao. “An intelligent vision-based approach for helmet identification for work safety”. *Computers in Industry* 100 (2018), pp. 267–277.
- [2] Nathan E. Sanders, Elif Şener, and Karen B. Chen. “Robot-related injuries in the workplace: An analysis of OSHA Severe Injury Reports”. *Applied Ergonomics* 121 (2024), p. 104324.
- [3] Muhammed Enes Atik, Zaide Duran, and Roni Özgünlük. “Comparison of YOLO Versions for Object Detection from Aerial Images”. *International Journal of Environment and Geoinformatics* 9.2 (2022), pp. 87–93.
- [4] Ibrahim Yousif et al. “Safety 4.0: Harnessing computer vision for advanced industrial protection”. *Manufacturing Letters* 41 (2024). 52nd SME North American Manufacturing Research Conference (NAMRC 52), pp. 1342–1356.
- [5] Yange Li et al. “Computer Vision-Based Hazard Identification of Construction Site Using Visual Relationship Detection and Ontology”. *Buildings* 12.6 (2022).
- [6] Sheila Anand and Loganathan Priya. *A Guide for Machine Vision in Quality Control*. Vol. 1. 1. Chapman and Hall/CRC, 2020, p. 208.
- [7] Bo Zhou et al. “Design and Implementation of Intelligent Security Robot Based on Lidar and Vision Fusion*[”]. *Journal of Physics: Conference Series* 2216.1 (Mar. 2022), p. 012013.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [9] Charu C. Aggarwal. *Neural networks and deep learning : a textbook*. eng. Springer, 2018.
- [10] Fahad Alrasheedi, Xin Zhong, and Pei-Chi Huang. *Padding Module: Learning the Padding in Deep Neural Networks*. 2023.
- [11] Guilin Liu et al. *Partial Convolution based Padding*. 2018.
- [12] Konstantinos G. Liakos et al. “Machine Learning in Agriculture: A Review”. *Sensors* 18.8 (2018).
- [13] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. “A Survey on Performance Metrics for Object-Detection Algorithms”. eng. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020, pp. 237–242.
- [14] Zhengxia Zou et al. “Object Detection in 20 Years: A Survey”. *Proceedings of the IEEE* 111.3 (2023), pp. 257–276.

- [15] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.
- [16] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 6517–6525.
- [17] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". *CoRR* abs/1804.02767 (2018).
- [18] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". (2020).
- [19] Chuyi Li et al. "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications". (2022).
- [20] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". (2022).
- [21] Dillon Reis et al. "Real-Time Flying Object Detection with YOLOv8". (2024).
- [22] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information". (2024).
- [23] Guangzhen Yao et al. "HP-YOLOv8: High-Precision Small Object Detection Algorithm for Remote Sensing Images". *Sensors* 24.15 (2024).
- [24] Shifeng Zhang et al. *Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection*. 2020.
- [25] Satya Mallick. *Yolo - learnopencv*. <https://learnopencv.com/yolo11/>.
- [26] Rahima Khanam and Muhammad Hussain. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024.
- [27] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361.
- [28] Narimane Wafaa Krolkral et al. "Improved YOLOv5s for Object Detection". *2023 International Conference on Electrical Engineering and Advanced Technology (ICEEAT)*. Vol. 1. 2023, pp. 1–6.
- [29] Sangjune Shin and Dongkun Shin. *Octave-YOLO: Cross frequency detection network with octave convolution*. 2024.
- [30] Wenhan Luo et al. "Multiple object tracking: A literature review". *Artificial Intelligence* 293 (Apr. 2021), p. 103448.
- [31] François Auger et al. "Industrial Applications of the Kalman Filter: A Review". *IEEE Transactions on Industrial Electronics* 60.12 (2013), pp. 5458–5471.
- [32] Zhenglin Wang, Kerry Walsh, and Anand Koirala. "Mango Fruit Load Estimation Using a Video Based MangoYOLO—Kalman Filter—Hungarian Algorithm Method". *Sensors* 19.12 (Feb. 2019).

- [33] Masoud Khodarahmi and Vafa Maihami. “A Review on Kalman Filter Models”. *Archives of Computational Methods in Engineering* 30.1 (Jan. 2023), pp. 727–747.
- [34] Greg Welch and Gary Bishop. “An Introduction to the Kalman Filter”. *Proc. Siggraph Course 8* (Jan. 2006).
- [35] H. W. Kuhn. “The Hungarian method for the assignment problem”. *Naval research logistics* 52.1 (2005), pp. 7–21.
- [36] T.H. Cormen et al. *Introduction to Algorithms, fourth edition*. MIT Press, 2022, p. 723.
- [37] Harold W. Kuhn. “A tale of three eras: The discovery and rediscovery of the Hungarian Method”. *European Journal of Operational Research* 219.3 (2012). Feature Clusters, pp. 641–651.
- [38] Atousa Zarindast and Anuj Sharma. “Opportunities and Challenges in Vehicle Tracking: A Computer Vision-Based Vehicle Tracking System”. *Data science for Transportation* 5.1 (2023).
- [39] Zheng Tang et al. “CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification”. (Mar. 2019).
- [40] Anton Milan et al. *MOT16: A Benchmark for Multi-Object Tracking*. 2016.
- [41] Yifu Zhang et al. “ByteTrack: Multi-Object Tracking by Associating Every Detection Box”. (2022).
- [42] Xuelin Qian et al. *Long-Term Cloth-Changing Person Re-identification*. 2020.
- [43] Wenyu Wei et al. “Person re-identification based on deep learning — An overview”. *Journal of Visual Communication and Image Representation* 82 (2022), p. 103418.
- [44] Mang Ye et al. “Deep Learning for Person Re-identification: A Survey and Outlook”. (2021).
- [45] Kaiming He et al. “Deep Residual Learning for Image Recognition”. (2015).
- [46] Liang Zheng et al. “Scalable Person Re-identification: A Benchmark”. eng. *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1116–1124.
- [47] Srikrishna Karanam et al. *A Systematic Evaluation and Benchmark for Person Re-Identification: Features, Metrics, and Datasets*. 2018.
- [48] Tong Xiao. *Evaluation Metrics*. https://cysu.github.io/open-reid/notes/evaluation_metrics.html.
- [49] Jakub Špaňhel et al. *Learning Feature Aggregation in Temporal Domain for Re-Identification*. 2019.
- [50] Yiru Zhao et al. “Attribute-Driven Feature Disentangling and Temporal Aggregation for Video Person Re-Identification”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4908–4917.
- [51] Debapriya Maji et al. *YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss*. 2022.

- [52] Sijie Yan, Yuanjun Xiong, and Dahua Lin. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*. 2018.